Host organization: School of Information Technology & Engineering (SITE) - University of Ottawa
800 Avenue King Edward
Ottawa, Ontario
K1N 6N5 Canada

Student: **Julien DAVID**

Design of a RFID reader controller

Final Report

Internship period	11th of February to 11th of July
Deepening module	Computer science & Network
Company Supervisor	Pr Miodrag Bolic : mbolic@site.uottawa.ca
ESISAR Supervisor	Dr Christophe Deleuze : christophe.deleuze@esisar.inpg.fr

Abstract

RFID is an acronym for Radio Frequency IDentification which is a wireless communication technology used to identify any sort of object which has a tag being attached to it. It enables an unique identification for all these objects that have been tagged, that's why it is more and more used in industrial applications such as supply chain management or asset tracking.

In a RFID network, with multiple readers, it is interesting to collect various information about reader's operations and tag reports. As the amount of connected readers can be very important (up to 100 readers), it is essential to automate this control.

The objective of this project is to build a central unit which would be able to control and synchronize the readers, process the information obtained from all the readers and configure specific scanning operations on the readers.

This report explains the approach done in the different steps of the project which are:

- Analyze and choose software for RFID Reader simulation. Because of the large amount of readers to control it was impossible to work only with real readers.
- Study the *Low Level Reader Protocol* (LLRP) in order to identify the main and essentials features.
- Implement the main LLRP operations on a basic controller
- Develop a software which is able to connect to several readers and allows the user to set up operations on all these readers.

But before going through these issues I will present the context of the project, with a detailed description of the main RFID component and the protocol LLRP.

There is a French synthesis of this technical report in appendix C.

Acknowledgements:

I would like to thanks Pr Miodrag Bolic for his support during this project : supplying me the resources of the University of Ottawa and guiding me in this project development.

Table of content

ABSTRACT **2**

TABLE OF CONTENT **3**

INTRODUCTION **5**

RFID SYSTEM 5

RFID Reader *8*

Tag *8*

LLRP 10

Vocabulary definitions *12*

CONTEXT PRESENTATION **14**

UNIVERSITY OF OTTAWA 14

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING 14

CONTEXT OF THE PROJECT 15

PROJECT SPECIFICATIONS **17**

PROJECT DEVELOPMENT **18**

COMPARATIVE ANALYSIS OF RFID SIMULATORS 18

Criterions *18*

Simulators *18*

Conclusion *24*

LLRP STUDIES 25

Reader Operation Section *26*

Reader Device Capabilities Section *28*

Reader Device Config Section *29*

CONTROLLER DEVELOPMENT 29

Introduction to LLRP : Basic controller for one reader *30*

Basic controller for two readers *32*

Complete controller for two readers or more *35*

FINANCIAL ESTIMATION **39**

CONCLUSION **40**

APPENDICES **41**

APPENDIX A - USER MANUAL 41

Graphical User Interface Presentation *41*

Setup a Reader Operation *42*

Setup the tag filtering *45*

APPENDIX B – SOURCE CODE DOCUMENTATION 46

APPENDIX C – SYNTHESIS IN FRENCH/SYNTHESE EN FRANÇAIS 48

Introduction *48*
Développement du projet *50*
APPENDIX D - RIFIDI EMULATOR DOCUMENTATION 56
APPENDIX E – EXAMPLES OF LLRP MESSAGES 59
GET_READER_CAPABILITIES_RESPONSE *59*
ADD_ROSPEC_MESSAGE *60*
RO_ACCESS_REPORT *61*
REFERENCES 64

Introduction

As the main domain of this project is the RFID, this introduction will present the basics knowledge about this technology. We will see quickly what is an RFID system, how does it work and what it consists in. After this first part I will present the main components of an RFID system, the readers and the tags and the dialog between these two entities.

Then I will present LLRP, the protocol used as interface between the readers and a client. Usually, is called *Client* the entity which dialog with a reader via LLRP, the client may be software, but it could be hardware too.

RFID System

RFID (Radio Frequency IDentification) is an auto-identification technology using radio frequencies signals. This is a powerful technology that may replace soon the bar codes. It is more and more used in the industry for various applications such as the supply chains management; identify supermarket goods and even now in the passports.

The key of this system is the dialog between the reader and the transponder (also called tag). The transponder stores information on a small memory which is powered by the magnetic field transmit by the reader. In a basic communication, the reader scan a tag population, that means it sends radiofrequency signals. Powered by this signal the tag sends back the content of its memory to the reader.

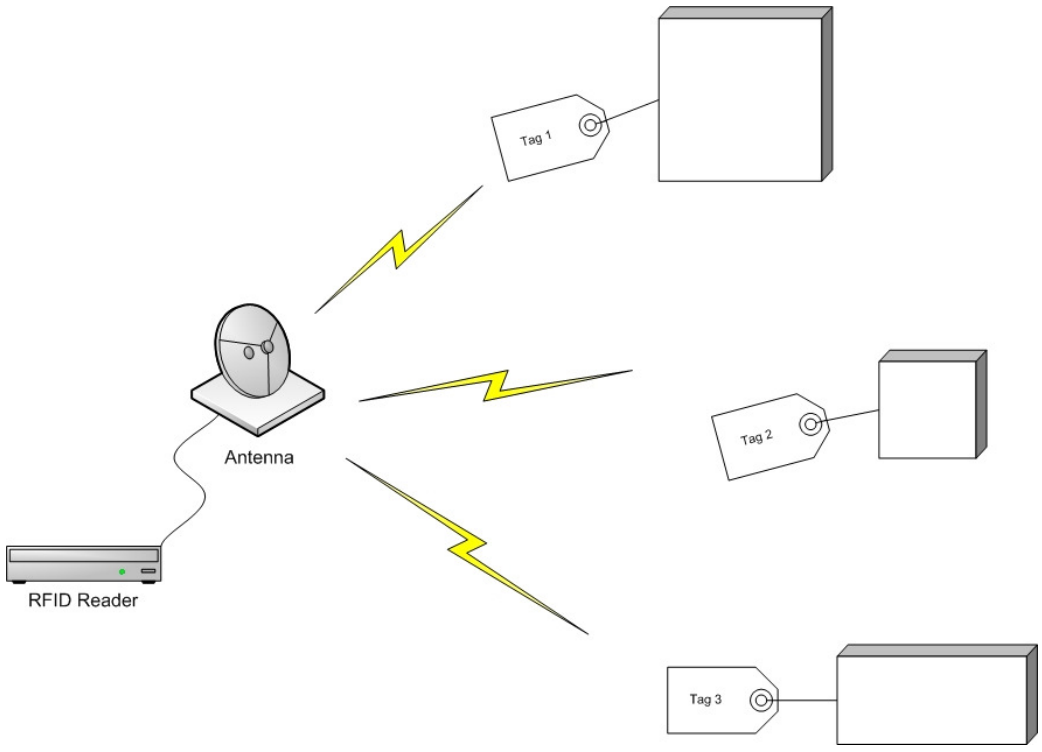


Figure 1 - RFID System

The figure 1 below shows how the reader sends the signal to power and queries the tags. It sends a signal whose magnetic field will power the tag. Once the tag is turned on it can received and decode the signal. As the signal contains a query for the tag, if the tag corresponds to the query it answers to the reader, if not it stays quiet. The following figure (fig2) explains the exchanges between the reader and the tags. In this example the reader use a filter, that means that it doesn't query all the tags but only those who corresponds to the specific criteria.

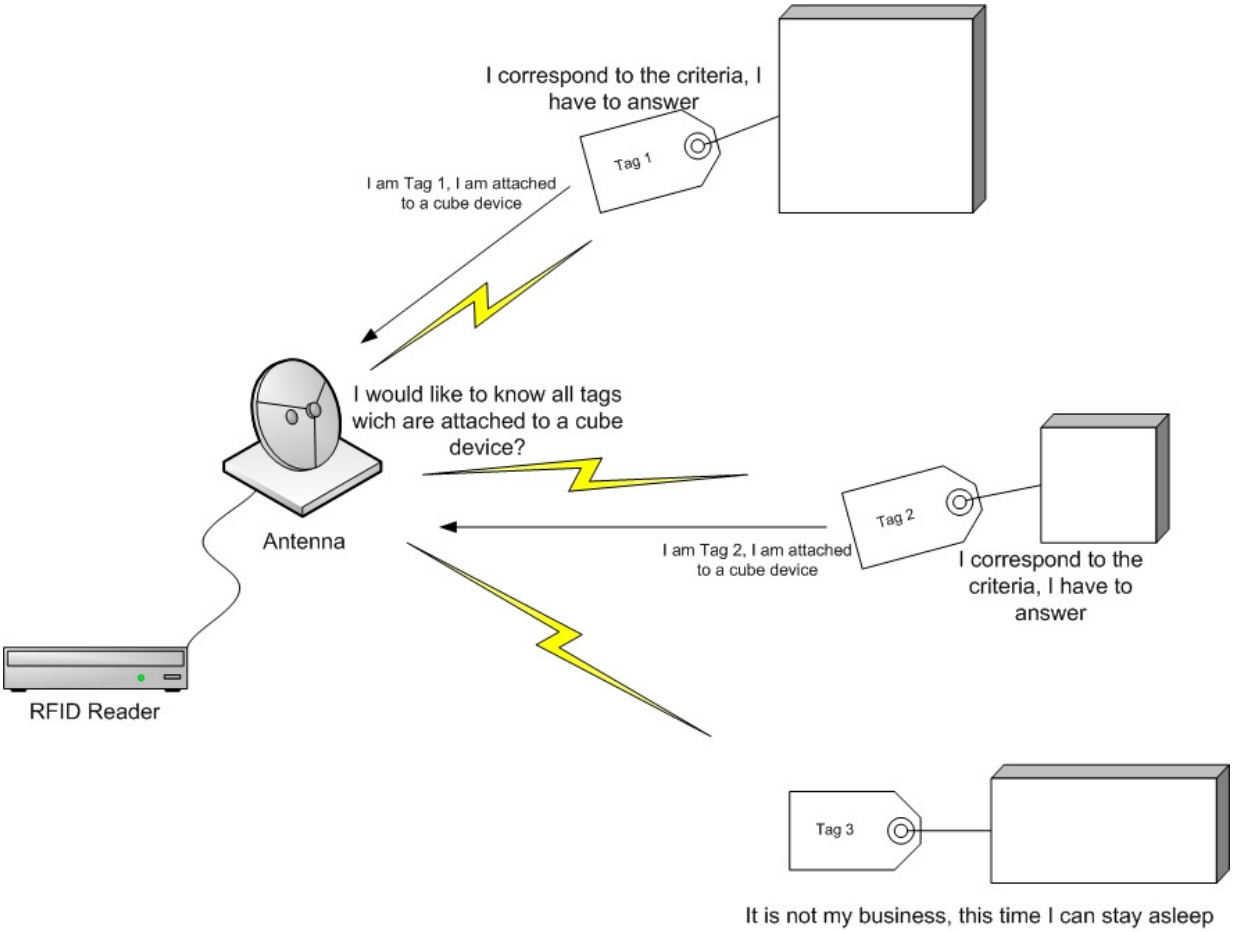


Figure 2 - Exchanges between the reader and the tags

Example of direct application

> *In a supermarket's warehouse, all the items here have a tag attached to identify them. If the director wants to know how many milk bottles he has in the warehouse, instead of sending a trainee counting all the bottles he will used his RFID reader to query all the tags with a particularity in their ID. If this particularity signifies that the tag is attached to a bottle of milk the director could have his answer quicker than with the trainee.*

This is of course a schematic view of a RFID system, actually it can't work like this because of the tag collision. Like in all wireless communication technology there is

a collision problem, that's mean that two entities can't "speak" at the same time. If two or more queried tags try to answer at the same time the reader won't be able to decode all the signals. The different answers may interfere each other and confuse the reader which won't be able to differentiate the signals. In order to get around this collision problem when the reader detects a tag collision, ie when it receives a signal that he can't decode, it sends a *Gap Pulse Signal*.

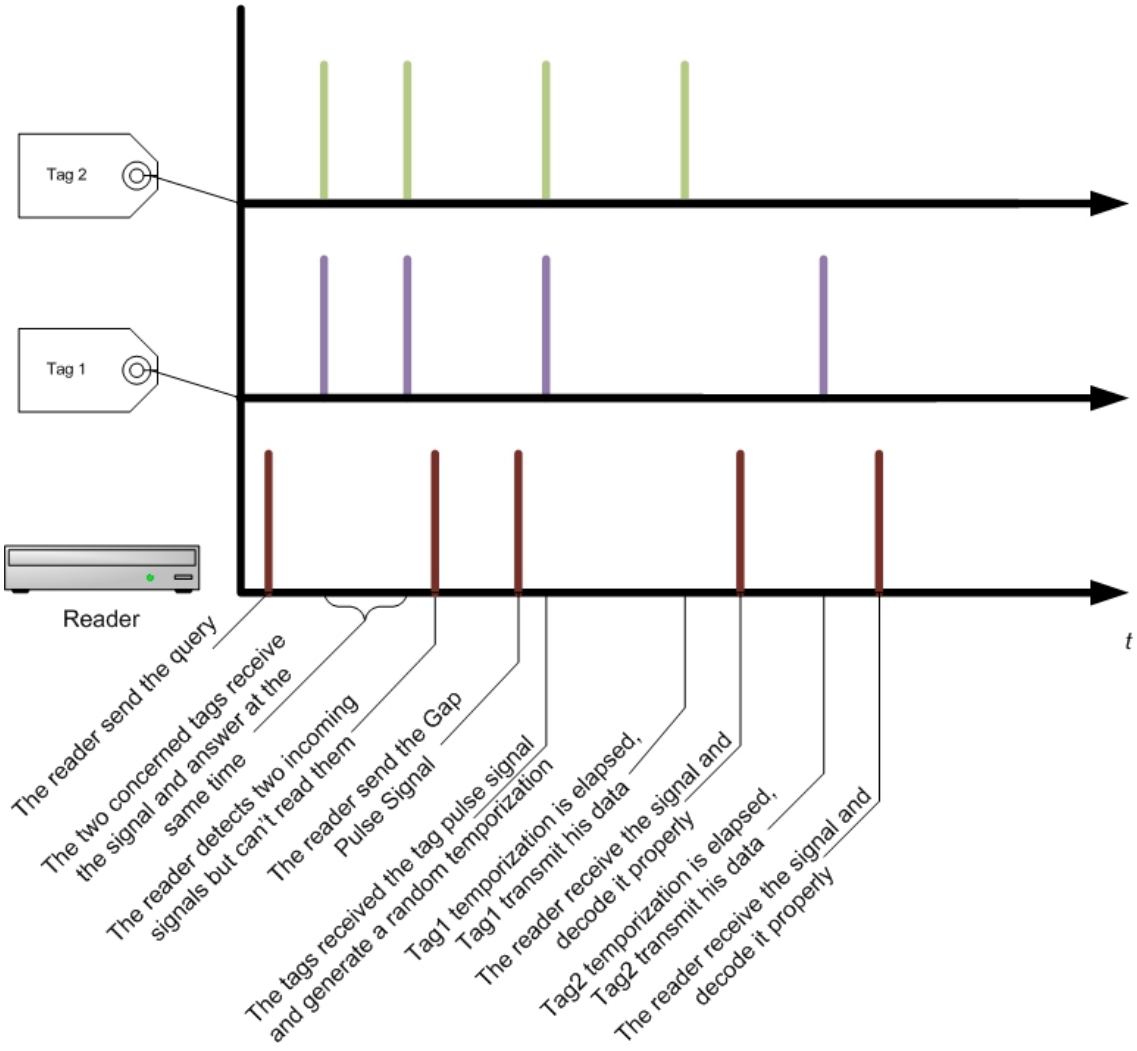


Figure 3 - Anti tag collision protocol

Once a tag receives this specific signal it generates a random temporization before resend his data. As all the queried tags receive the signal and they all generate the random temporization they won't answer at the same time and the reader will be able to decode the different answers. The figure above (fig 3) shows this sequence.

This was a quick overview about the RFID technology and a simple anti tag collision algorithm. The following section will detail the two main components: the readers and the tags.

RFID Reader

RFID readers or interrogators are devices which have the capability of reading the information stored in the RFID tags but also writing data to them. This act of writing is called creating a tag, and along with associating it with an object to the tag is called commissioning the tag. In terms of passive tags they also have to power the tag.

Readers are roughly made of the following 2 components:

- Antenna
- RF Electronic Modules

A reader can have one to four antennas connected. The antenna is the device which sends the signal, so this is the part who dialog with the tag. There are several types of antenna depending on the application.

The RF module is the *"big box"* of the reader, this is the electronic and process part of the reader. All the antennas are connected to the RF module but also the serial link or the network link.

Readers are usually classified into two different groups:

- Serial reader
- Network reader

The **serial readers** use a serial communication link to communicate with an application and usually serially connected to a computer through RS-232 or RS-485 serial connections. The main advantage of this sort of readers is reliability of communication link and as the result using these readers is minimizing the dependency on a communication channel. Their main disadvantage is the dependency on maximum cable length that can be used to connect them to a computer.

The **network readers** communicate with an application through both wired and wireless communication. Unlike their serial counterparts they don't depend on the maximum cable length that can be used to connect them to a computer.

Another advantage that they have is the ability to update their firmware remotely without any need to physically visit the reader and consequently the maintenance effort and ownership costs of these readers are lower. The disadvantage they have when compared to their serial counterpart is that they are not as reliable as serial readers.

Tag

The tags are the other part of the RFID system, this is a small component composed essentially by an antenna and an integrated circuit. This circuit contains a small readable and writable memory to store the information and the antenna is used to receive and transmit the signals from and to the reader.

The two mains categories of tags are the passive tags and the active tags. The difference between these two categories are the presence or not of a battery as power source. But as their working is the same, we will discuss first about the commons parts.

Both of these two categories of tags are composed by a memory. This memory is composed by four parts(fig4) [1] : a reserved part for the password used for the lock or kill functions, the EPC part which contains a unique identifier, generally the EPC, the third part contains information about the tag, possibly including a unique tag identifier distinct from the EPC and the last section of the memory is called user memory and is organized as the user wants.

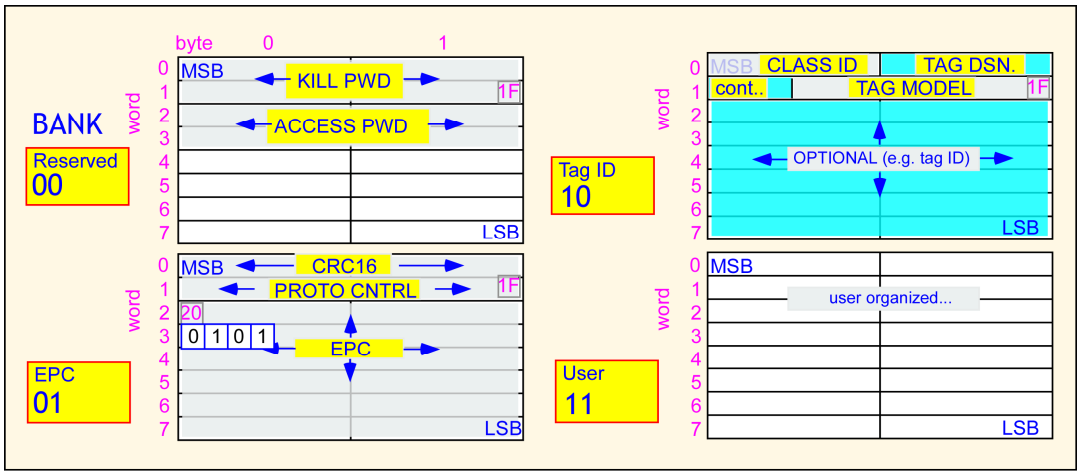


Figure 4 - Tag memory decomposition [1]

The passives tags are those which do not have battery: they use power induced by the received signal to power up the circuit and transmit the answer. That's why a dialog with a passive tag must be initialized by the reader. As the result passive tags are simpler to make and due to this simplicity their manufacturing cost is lower and subsequently the range in which they can be either interrogated by the reader or their signal be transmitted to the reader is much shorter compare to other tags types and is usually between 2,5cm to about 100m [2].

Active tags are those which are carrying an onboard power source although this shouldn't necessarily be a battery source as other sources such as solar might be feasible as well. Two different terms are often used to express the type of active tags, they are transmitter and transponder. Transmitters are those kinds of active tags which transmit data all the time regardless of presence or absence of a reader because it is not necessary for the tag to have a reader in accessible range. Transponders on the other hand are those which can be entered into a sleep state when there is no reader in their accessible range and as the result they are capable of having a much longer lifetime as they are waiting for a wake up command from the reader to start up and therefore saving battery on other occasions.

The batteries in active tags can last from two to seven years. The onboard battery feature makes active tags more expensive and complex and also larger to make

compare to their passive counterparts but also makes them work in a much larger range as well as the capability of working in much higher frequencies.

Although active and passive are two different types of the same component (RFID tag), usually depending on which type of tag has been used, these two different names are also given to the whole systems which lead to Active RFID systems and Passive RFID systems.

These two technologies while usually considered and evaluated together, are fundamentally distinct technologies with different capabilities. Therefore the most complete solutions use the advantages of both technologies and combine them in complementary ways. During the development of the project we use only passives tags, because of their low cost.

LLRP

The EPC Global [3] ratified in April 2007 the Low Level Reader Protocol [4]. This protocol is a standardization of the network interface for the RFID Readers. After having standardized the tag and reader radio frequency air interface protocol, this LLRP, as the logical next step, was one of the most waited in the RFID domain, it was considered by the professional as the missing level to promote RFID technology and to promote it to a high using level.

Besides offering to common network interface to all types of readers, LLRP also provides full access to all features of the air protocols supported by the reader, that's why this protocol is called *low-level*.

The design of this interface recognizes that in some RFID systems, there is a requirement for explicit knowledge of RFID air protocols and the ability to control Readers that implement RFID air protocol communications. It also recognizes that coupling control to the physical layers of an RFID infrastructure may be useful for the purpose of mitigating RFID interference.

LLRP provides some functions used by a Client to connect to the Reader in order to perform operations. It is specifically concerned with providing the formats and procedures of communications between the two RFID entities. The LLRP protocol data units are called messages. Appendix E illustrates with 3 examples the format of data exchanges. Messages from the Client to the Reader include getting and setting configuration of Readers, capabilities discovery of Readers and managing the inventory and access operations at the Readers. Messages from the Reader to the Client include the reporting of Reader status, RF survey, and inventory and access results. The figure 5 below shows how these messages take place in a typical LLRP sequence.

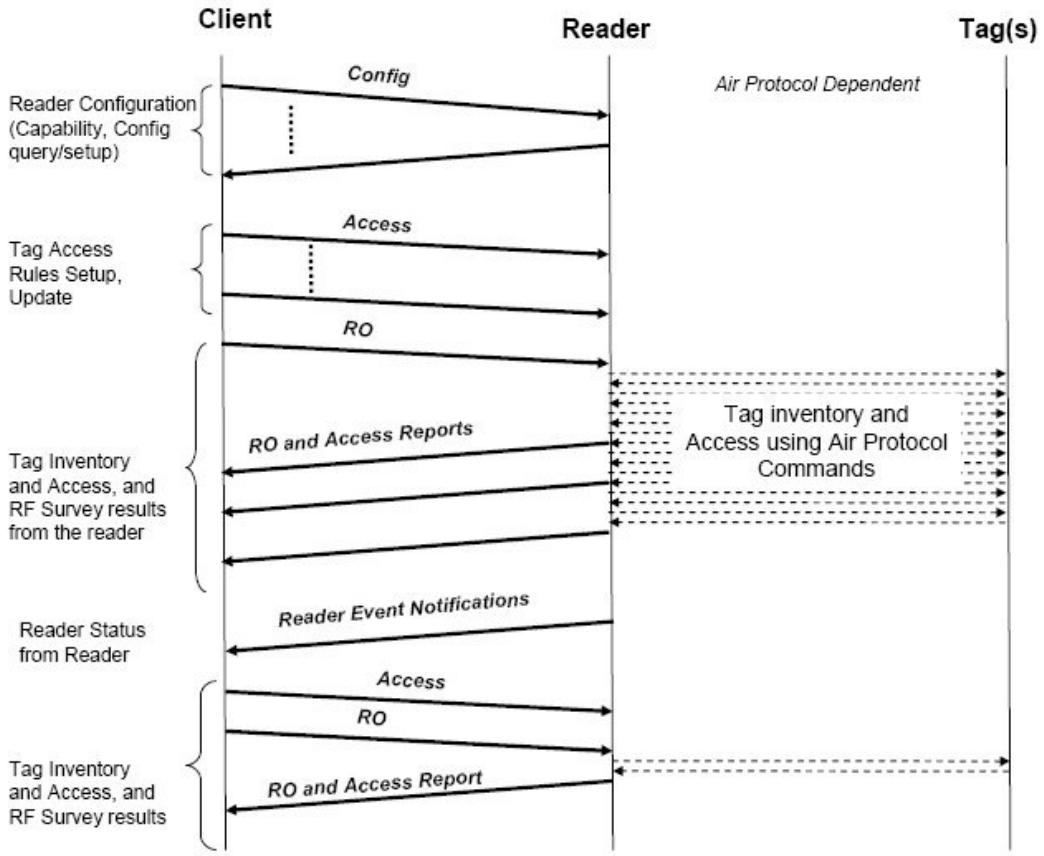


Figure 5 - Typical LLRP Timeline

LLRP is an application layer protocol and does not provide retransmission, or reordering facilities. State consistency between the Client and the Reader is critical for the correct functioning of the system. Using LLRP messages, the Client updates the Reader state which includes Reader configuration parameters, dynamically created data structures (ROSpecs…), and possibly vendor defined data. For this reason, LLRP requires acknowledgements for the Client to Reader transactions – this provides a fail-safe mechanism at the LLRP layer to cope with network error situations. Also, to cope with intermittent connections, a Client can request a Reader's configuration state to confirm that a Reader's state is consistent with the Client after the Client reconnects.

LLRP is based upon an abstraction of RFID air protocols and their respective commands. There are two principal concepts to the LLRP abstraction of RF operations by a Reader: Reader Operations (RO) and Access operations (AO). Reader Operations (RO) define the parameters for operations such as Antenna Inventory and RF Survey, Access Operations define the parameters for performing data access operations to and from a tag.

The timing control of an operation is specified using boundary specification, which specifies how the beginning (using start trigger) and the end (using stop trigger) of the operation are to be determined.

This is a basic presentation of LLRP with the main concepts, the following section will detail two points. First the Client and then the Reader Operations which is the most used operation in this controller.

Vocabulary definitions

Client

As shown on the figure 6, LLRP has two end-points: the Reader end-point and the Client end-point. The LLRP workgroup deliberated the naming of the Client end-point. In the first draft of the LLRP spec, the Client end-point was, instead, called the "Host" end-point. After many discussions this term was abandoned because of the familiarity to the IT. Another proposition, "Controller", was also cleared because it would imply that this end-point would be hardware, which is not the aim of the developers.

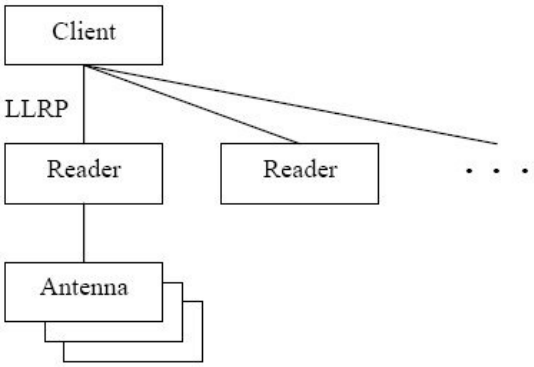


Figure 6 - LLRP Endpoints

As a compromise, now in the LLRP spec it's called the "Client" end-point. So, anything that talks to a LLRP RFID Reader is simply called a "Client". This naming will be used to indicate the controller.

Reader Operation

The reader operations (RO) are one of the types of operation that a Client can set up on a reader. As the other type of operation is not implemented in the controller it won't be detailed in this section.

A reader operation defines the parameters for the operations that will be executed by the reader. It contains at least one spec, this spec could be an Antenna Inventory Spec (AISpec) or a RFSurveySpec.

An AISpec defines an inventory operation to be executed by the reader, that means that the reader will query all the tags present in his read range. The other kind of operation, RFSurvey operation, is an operation during which the reader performs a scan and measures the power levels across a set of frequencies at an antenna.

Many RO can be set up on a reader, depending of the reader capabilities, each one with his priority. If all the operations have the same priority, the first started is executed and as soon as the reader has a free slot, the second one can be launched too. However some readers allow only a restricted number of configured RO even if they are not started.

Context presentation

University of Ottawa

The internship took place in a lab at the University of Ottawa, particularly in the SITE (School of Information Technology & Engineering) department [5].

Located in Ottawa downtown, Ottawa University is a bilingual university, that means that the courses are fairly divided both in English and in French. Founded in 1848 by the Missionary Oblates of Mary Immaculate, the Bytown College was at the beginning a small catholic school [6]. Now with more than 30,000 students in 11 different faculties Ottawa University is the most important bilingual university of North America.

The university purposes courses of first, second and third level and professional programs in a large range of disciplines in the eleven faculties:

- Art
- Civil law
- Common law
- Education
- Engineering
- Graduate and Postdoctoral studies
- Health sciences
- Medicine
- Sciences
- Social sciences
- Telfer school of management

The research is also now an important activity for the university because it is part of the *Group of Thirteen*, which is a group that describes itself as the leading research-intensive universities in Canada. But according to the RE$EARCH Infosource 2007 publication [7], the university is ranking as 5th in research intensity and 9th in total research funding.

School of Information Technology and Engineering

The School of Information Technology and Engineering (SITE) is the University of Ottawa's centre for research and teaching in all domains related to computing, computers and communications. This department founded in 1997 by the merger of the Department of Computer Science and the Department of Electrical and Computer Engineering, counts now about 775 undergraduates students and 400

undergraduates students working on advanced degrees on Computer Science and Electrical and Computer Engineering.

This department is also very active in research with over than 70 permanents researchers working with the students and leading the research groups. There are 15 research groups based on Photonics, Software Engineering, Robotics, Bioinformatics and Biomedical Engineering or Wireless Networks and Mobile Computing in which this project was attached.

SITE has also strong connections with local industry and this results in a dynamic environment of collaboration and practical impact. Many research projects are entrusted by these industries to various labs of the university and many undergraduates students after working for a company during their studies are employed by the companies.

The project beneficed of all the resources available at the SITE, computers of course but also RFID components (readers and tags).

Context of the project

This RFID reader controller project is part of a big research proposal. Initiated by four professors from the University of Ottawa, Prs Bolic, Khan, Yagoub and Yang, this project is called "*Development and Simulation of Enhanced RFID Systems*" [8]. The objectives of this research project are multiple, the first one is to get important knowledge about RFID. Because of its high potential to improve, automate and facilitate many existing applications, RFID is really a key technology and it is really important for researchers to master it.

This first objective is more theoretical than the others and the second objective is to develop a new RFID reader with uncommon features that would be used for several new applications. These features include detection of fast moving tags, positioning of tags or mitigation of interference problems. Of course this reader development would go with the development of all the others RFID components, such as new signal processing algorithms or new antenna design.

The third objective of the project is to develop an RFID deployment simulator. After identifying the main weaknesses of the RFID it appears that they include erroneous reads, problems of handling of large amount of data generated by the tags, interferences problems and networking problems. Build a complete RFID simulator would be helpful to identify the source of these weaknesses and correct them.

This simulator is divided in five parts and one of them is "*A baseline RFID simulator*". This part includes a 3D simulator for reader to tag communication and the simulator for controlling multiples readers. Another student from ESISAR, Guillaume Chanas, worked on the 3D simulator project and my own contribution has been focused on the reader controller. As the final motivation is to develop a deployment 3D

simulator that would includes these two parts the two simulators will converge, but these two projects did not reach yet the point where the collaboration between us would be mandatory.

Project specifications

In this project, RFID reader controller will be implemented. This controller will be able to control up to 100 RFID readers and to process information obtained from tags. The project has several phases:

1. Developing a simulator of a system
2. Propose the simulation environment – this can be a stock in which RFID readers will scan the population of tags every 5min and report to the upper layers if there are any changes in the tag population. The system must be able to break the regular reading procedure and to perform queries of specific items (One processor might be allocated to do that)
3. Determining the network topology and the number of routers needed

The controller will be able to perform the following:

- **Reader control and coordination**

We will work with the grid of readers that have one or four antennas. For readers that have one antenna, only the non-neighboring readers will perform query at the same time.

For the neighboring readers many solutions can be implemented, the easiest is time-multiplexing, but other could be studied during the project.

- **Tag data smoothing and filtering**

Filters will be set by higher application layers. Some of the filters include: detecting tags that are read by two or more readers, detecting tags that are constantly read by the same reader, detecting moving tags and so on.

The EPC Global C1G2 filtering functions will also be implemented, this is a filter which works on the tag ID, only the tags which tag ID matches with the mask are read.

- **Reader-controller interface**

Readers and the controller are connected either using Ethernet or IEEE802.11 protocols. RFID standard used for communication is Low Level Reader Protocol (LLRP) [4].

- **Reader-tag interface**

We will focus on EPC Global Class 1 Gen 2 UHF standard [9], because it is the commonly used.

Project development

The project has been divided in three distinct parts, the first thing to do was study the different RFID simulators and choose one. Then I started working on LLRP, study the protocol and spot the essentials features. At last I worked on the development of a basic controller and then on the final controller.

Note: *There are in this part some references to the performances of the computer. My evaluation is based on the computer I used during this project, PC with 1,4 GHz processor 2Go of RAM equipped with WinXP.*

Comparative analysis of RFID simulators

This part has to present the different solutions for simulating RFID readers. First, the criterions for selection will be presented and explained. All this criterions are linked to the project specifications and has been defined regarding to these specifications.

Then all the simulators will be described with their strengths and weaknesses and finally all the solutions and how they correspond with the criterions will be summarized.

Criterions

The following criterions have been defined:

- Simulation of real readers
- Support of the Reader to Tag communication
- Support of the Reader to Tag protocol :
    - Support of EPC Global Class 1 Gen 2
- Support of LLRP
- Limit for the number of readers to simulate (the objective is to simulate 100s of readers)

The simulation of the dialog between the tags and the readers is required because some information are exchanged in this communication, for example the tag filtering is set up on the reader, and then the reader filter the tag using EPC Global C1G2. As tag filtering is one of the requirements for the controller this dialog has to be simulated.

Simulators

The various software studied are:

- RIFIDI
- Tag Centric
- Opnet
- RFID Anywhere

This list has been defined at the beginning of the project, they are all software that have been identified as RFID simulators.

RIFIDI

Here is the presentation of the software on the website [10] :

> Rifidi is the premier open source IDE for RFID. Rifidi let's you develop an RFID system entirely with Software Components and removes the dependency on hardware and infrastructure that RFID typically demands.
>
> Rifidi makes it possible to 'Virtualize' your infrastructure with Software Defined RFID Readers, RFID Tags, and RFID Events that behave like their real-life counterparts.

RIFIDI is an open source software, developed in JAVA. The source code is available via the SVN repository on the web [11]. This RIFIDI development project born when a team of industrial engineers tried to implement a client for 10's a readers and they were stuck by the complexity of this task. After discussing with many software development companies and RFID expert they realized that a RFID simulation tool would be appreciable for testing applications. Since that, in March 2006, this team of developers and RFID consultants started to work on this RFID simulation project.

The RIFIDI project contains two parts, RIFIDI Designer and RIFIDI Emulator. They are two different software developed in parallel by the same team.

The Designer tool is used to visualize an RFID system. The user can set up his production chain with the conveyors, position a reader gate and see how the tagged items behave in this environment (fig 7).

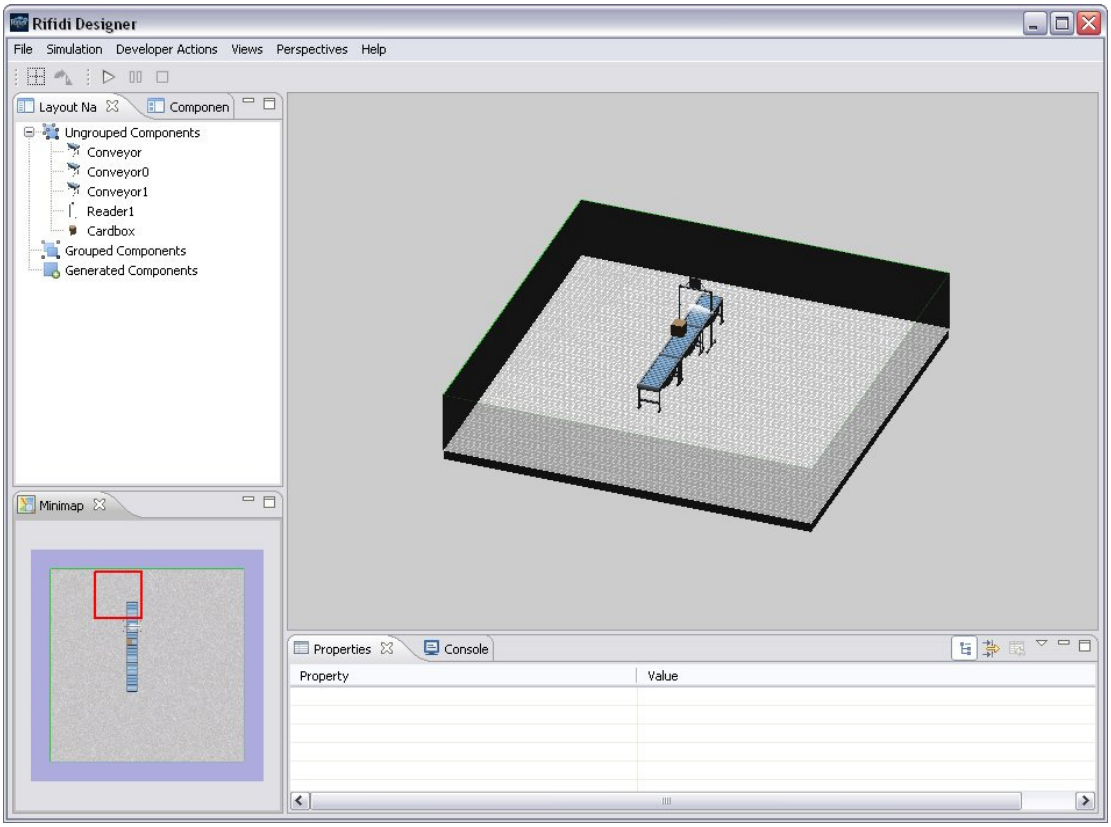


Figure 7 - RIFIDI Designer : A conveyor simulation

The main weakness of this simulator is his capacity, because of the GUI, which requires a lot of resources, the simulation capacity in terms of number of readers is very limited, only two readers can be simulated at the same.

The second part of the RIFIDI project is closer to that we need for this project. The Emulator tool is not far away from his small brother, the simulator engine is the same. But here there is no evolving environment, the user can just set up readers, generate tags and put them in the readers' read range, then remove, or change their area (fig 8).

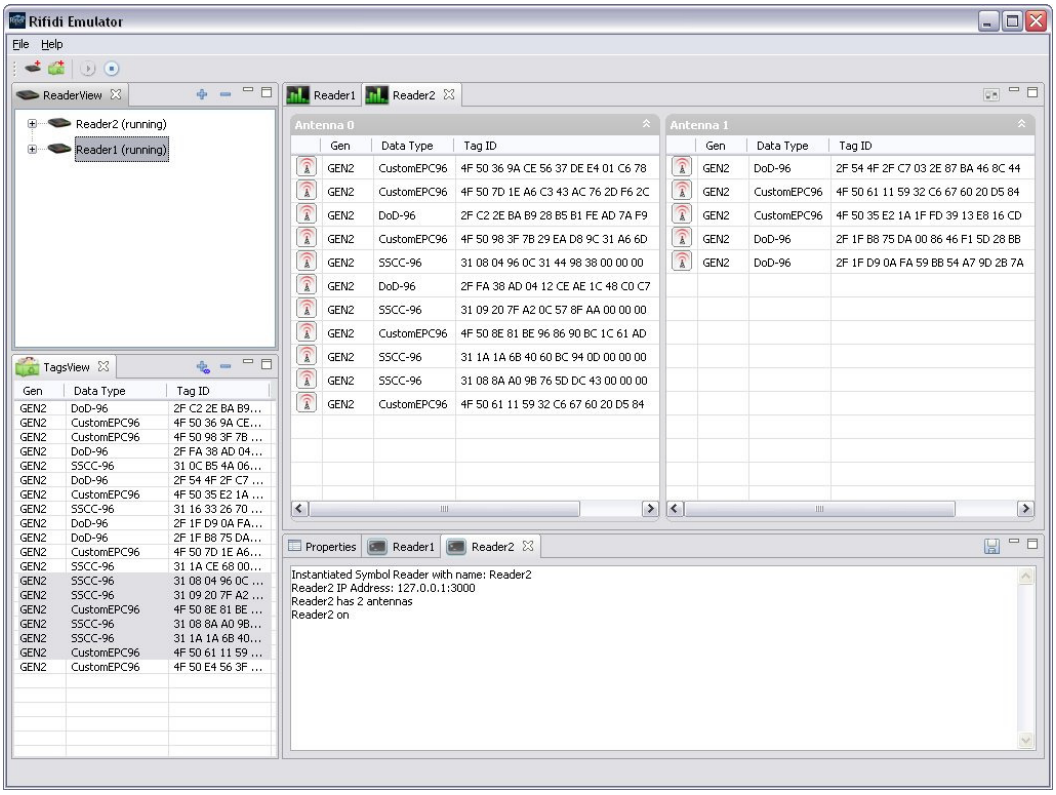


Figure 8 - RIFIDI Emulator : Simulating two readers

At the same time, using a console he can connect to the simulated reader to obtain reports about the activity of the reader. Specific reports of each tag are available and they contain much information, such as the *First Seen date*, *Seen Count* or the amount of power received (*Peak RSSI*).

All the reports are really a good point for this project, this is what we need, reports about the tags. Another advantage of RIFIDI Emulator is the support of real off-the shelf readers, there are six different readers to be simulated, two are fakes and four are real. Among the two fakes readers, one is called *LLRP Reader*, this one support the LLRP Protocol that we will use.

The black point of this emulator is, one more time the simulation capacity. Even without the graphical evolution of the environment, the limit is about six simulated readers running at the same time. Even if it is in progress regarding RIFIDI Designer it is still not enough for the project requirements.

However using a such user-friendly software for simulating only a couple of readers during the debugging step could be useful. As the only bad point is the amount of readers that can be simulated it is interesting to keep an eyes on RIFIDI Emulator.

Moreover the development team is very active, there are often new releases, it is available on an IRC channel and is listening all the incoming comments or demands.

TagCentric

This software, developed by a lab of the University of Arkansas (lab accredited by the EPC Global for RFID testing), is also an Open Source software. Written in Java language it is built as a collection of classes that wrap RFID related devices. It is a very simple software, with a user-friendly GUI.

> The general goal of this software effort is to provide a simple, open-source application capable of integrating RFID readers, RFID tag printers, and a database of the user's choosing. In the never-ending battle between simplicity and complexity, we generally err on the side of simplicity. Simple software is easier to manage, and simple user interfaces are easier for users to grasp.

This software is a not really a RFID Simulator. It allows the user to connect a database to his reader's network, in order to stock and study information about read tags. It is a kind of reader controller.

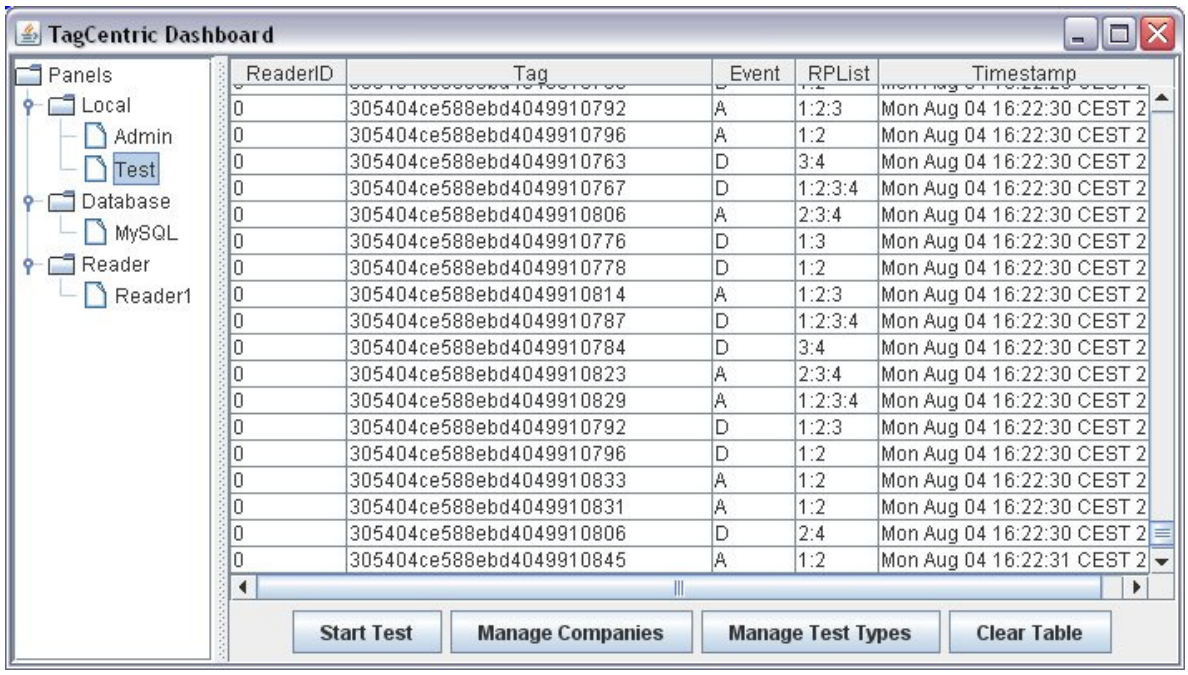


Figure 9 - Fake reader running on TagCentric

As it is possible to connect a fake reader (fig 9), which allows the user to have a virtual reader, we could use it for our simulation but the possibilities of this reader are very limited.

However, as the source code is available we could study it for inspiring us for the way to build our own controller.

RFID Anywhere

The third solution is, in my mind, the most complete. This is very big commercial software in which RFID simulation is only a part.

> RFID Anywhere is a flexible software infrastructure that integrates business logic and processes with a variety of automatic data collection and sensor technologies, including RFID, barcodes, mobile devices, PLCs, location tracking systems, environmental sensors and feedback mechanisms. This powerful infrastructure enables the creation of intelligent sensor networks out-of-the-box. By using RFID Anywhere, sensors are able to work together as an intelligent network by combining, organizing and coordinating these technologies through a common management structure, advanced feature set and event-driven development framework.[5]

I have downloaded the evaluation version and installed it. But as it is huge software I do not manage to really use it. After playing a bit with it, it appears to be adapted for us. It can simulate many protocols, not only LLRP, but almost all protocols existing on the different network layers.

There is specific module called *RFID Network Simulator*, this is really a physical lab for testing business logic. Everything can be simulated and tested, from the appearance to the departure of a set of tags. Even the tags can be modified to change the formats, the values, the used protocols… But we can also plug real devices to make a network using the simulated readers and the existing ones.

This is the opposite of RIFIDI, as the first one is specific, light and easy to use, RFID Anywhere is complex and large purpose. Of course it could simulate our RFID Network, because it can simulate everything in RFID, but it would take too much time to learn how to use it and we have to keep in mind that simulates the RFID reader network is only the introduction of the project.

RFID Anywhere would match with all the criterions but it is definitely to complex.

OPNET Modeler

I do not found technical documentation about OPNET Modeler, but regarding to the commercial documentation Opnet Modeler is a software used for all kind all network simulation.

Based on the simulation on the Multi-Channel MAC protocol for RFID Readers [12] the software is supposed to be able to simulate at least twenty five readers, this is also possible to simulate reader to tag protocols and to have an Ethernet channel or other wireless channel. This software is often used to validate protocols, such as RFID readers to tag protocols but also reader to reader protocols (Colorwave protocol for example).

The main advantage is the variable abstraction level, the user can specify a network using existing device in the library but he can also go to a lower level in order to modify the behavior of each component of the network. The level can be downed to

the point where the user can create a new device from a finites states machine definition.

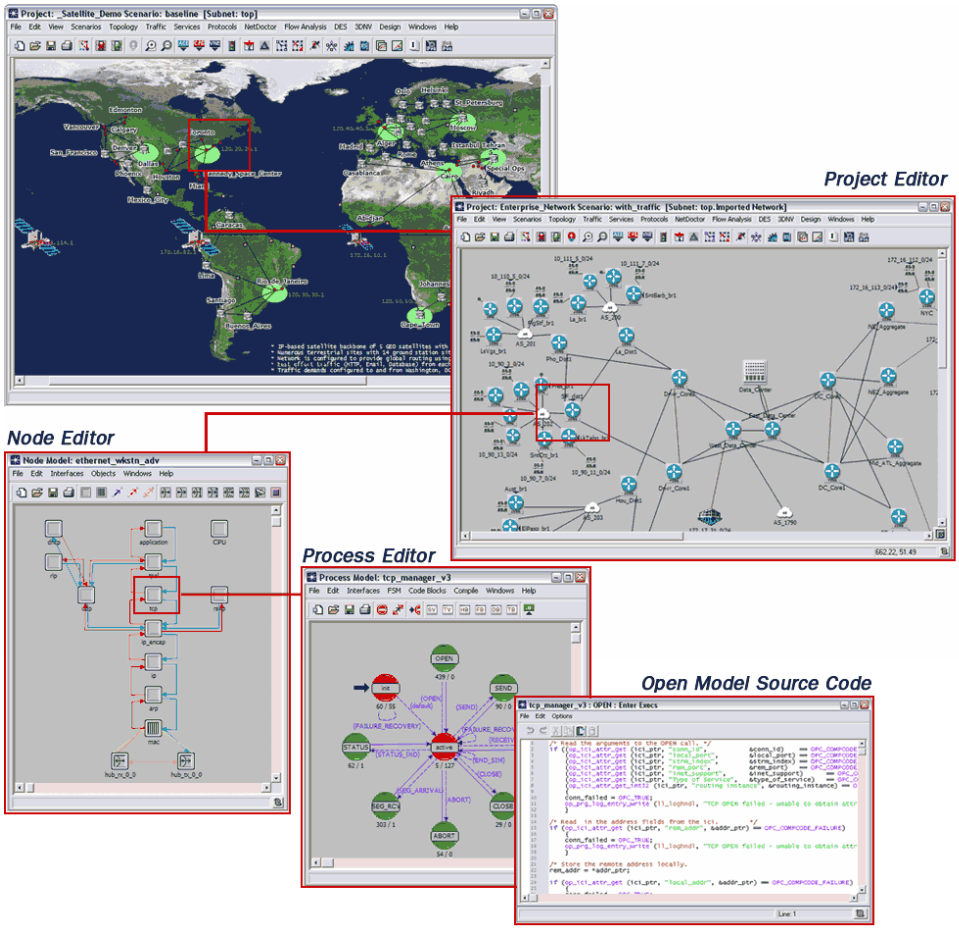


Figure 10 - All abstraction levels available in Opnet Modeler[13]

All levels can be modified in order to simulate precisely a network (fig 10). After this first step of network definition the user can specify all the data he wants to analyze.

Because of the flexibility of the simulation Opnet modeler is really a software that matches to our project needs. It supports LLRP, we can simulate a large amount of nodes and we can modify the behavior of each node to simulate different kind of readers.

Conclusion

Table in figure 11 recalls the main characteristics of the analyzed software.

Tag Centric is definitively not adapted for the project, as I said in its presentation it could be interesting to see how it is build but we can't use it in this project.

I also eliminated RIFIDI Designer because it doesn't correspond to the use we need, this software has been created to simulate the use of RFID in the production chain, not to simulate a large amount of readers.

It is a powerful system, but we don't need to visualize how the system evolves, at least for the development step. Perhaps, in the next phase of the project it would be useful to have an evolving environment, but for the moment, a fixed one is enough.

Concerning RIFIDI Emulator the number of simulated reader is also a discriminating criterion, but all the others criterions match with our needs.

OPNET seems to be the most useful program for the project, all the criterions corresponds to that we need. However, this is also a complex program, and as we don't need to simulate all the readers during the debugging step RIFIDI Emulator will also be used.

These two software match to all the requirements, we will use RIFIDI when only few readers will be required and OPNET when the controller will be build and when a large amount of readers will be needed.

Criterion / Software	RIFIDI Emulator	Tag Centric	RFID Anywhere	Opnet Modeler
Real readers	Yes	No	Yes	Yes
Reader to tag communication	Yes	Yes	Na	Yes
Reader to tag Protocol	Yes	No	Yes	Yes
EPC Global C1G2	Yes	No	No	Yes
LLRP	Yes	No	Yes	Yes
Max reader	6	3	Na	At least 25

Figure 11 - Simulators comparison table

LLRP Studies

This part presents the different features of LLRP that have been implemented in the controller. There are four sections, with a description of each function and why it is essential in the controller. The basics functions for the connection to the reader or the

disconnection and all their parameters are not presented here because they are so evident.

Reader Operation Section

This section provides functions to set up and manage the reader operations.

ADD_ROSPEC

This message is used to create a Reader Operation on the reader. Although LLRP supports configuration of multiple ROSpec in the same message this features won't be used, only one operation will be set up in each message.

This message contains all parameters for the operation, such as the start trigger type, stop trigger type, operation type or reports configuration. As a ROSpec is identified by a ROSpecID this message must specify this ID, the identifier is generated by the client when it requests the creation of the operation.

Receiving this message, the reader creates the operation in the state specified in the parameters. Actually this is not really a parameter because only the waiting state is accepted, a message must not accept an ADD_ROSPEC Message where the starting state would not be *Disabled*. The following step to activate the operation is the ENABLE_ROSPEC Message which put the operation in the *Inactive* state.

This function has to be implemented even for a basic use of LLRP.

ADD_ROSPEC_RESPONSE

This is the message sent by the reader after an ADD_ROSPEC message, if there is a problem creating the operation, the error code is included in this response.

If the ROSPEC is successfully created the success code is returned in the LLRPStatus parameter.

DELETE_ROSPEC

This command is used to delete an existing ROSpec on the reader. The ROSpec identifier must be precised in the parameter. It is important to notice that the identifier 0 corresponds to the deletion of all the existing ROSpec on the reader.

A ROSpec can be delete at anytime, no matter of the state, even if it is active, a DELETE_ROSPEC message stop it and then delete it.

Even if it is not a mandatory function it is very useful to clean the reader's configuration and quite easy to implement because of the low number of parameters.

DELETE_ROSPEC_RESPONSE

This is the response of the DELETE_ROSPEC command, it contains the status of the operation in the LLRPStatus parameter.

If the deletion has not been done, for any reason, the corresponding error code indicates the cause in the LLRPStatus parameter.

START_ROSPEC

This message is sent by the client to the reader. It asks the reader to put an operation in the *Active state*.

The ROSpecID must be specified in parameter. A ROSpec must be in the *Inactive state* to be started.

START_ROSPEC_RESPONSE

This is the response sent back by the reader when receiving a START_ROSpec request. It contains the status of the operation in the LLRPStatus parameter.

The different failure causes are:

- There is not any operation with the specified id
- The operation is not in the *Inactive State*
- The maximum number of actives operations set up on the reader is reached

STOP_ROSPEC

This message, send by the client to the reader, stops the execution of the ROSpec corresponding to the ROSpecID passed in parameter. Upon receiving the message, the reader overrides all the priorities, stops the execution of the operation and moves the ROSpec to the *Inactive state*. But the operation is not deleted, it is still configured on the reader and can be launched once again with a START_ROSPEC message.

STOP_ROSPEC_RESPONSE

This is the response send by the reader when receiving a STOP_ROSPEC message. If the specified id does not correspond to an existing operation on the reader the LLRPStatus parameter will indicates it.

If the stop command has been run successfully the status code value is *M_Success*.

ENABLE_ROSPEC

This is the message send by the client to the reader used to move the ROSpec corresponding to the ROSpecID passed in parameter from the *Disable State* to the *Inactive State*.

ENABLE_ROSPEC_RESPONSE

This is the response by the Reader to a ENABLE_ROSPEC command. If there was a ROSpec corresponding to the ROSpecID, and the Reader was able to enable that ROSpec, then the success code is returned in the LLRPStatus parameter. If there is an error, the appropriate error code is returned in the LLRPStatus parameter.

GET_ROSPECS

This is the request from the Client to the Reader to retrieve all the ROSpecs that have been configured at the Reader.

This message is used to synchronize the controller with the reader. The controller connects to the reader and it doesn't know the state of the reader. Using this command allows to know the operations, and their characteristics, already configured.

GET_ROSPECS_RESPONSE

This is the response send back by the reader to the GET_ROSPECS request. The reader sends a list of all operations, no matter about their state.

This list contains all the ROSpec with the characteristics defined when creating it. The current state is also sent with the report parameters.

Reader Device Capabilities Section

This section is about functions used to get the capabilities of the reader.

GET_READER_CAPABILITIES

This message is a request send by the client to the reader. The response, detailed in the next paragraph, contains information about the reader and its capabilities.

GET_READER_CAPABILITIES_RESPONSE

This is the response to the GET_READER_CAPABILITIES request. By sending this message the reader informs the client about its capabilities.

The information send by the reader are grouped in four categories, GeneralDeviceCapabilities, LLRPCapabilities, RegulatoryCapabilities and AirProtocolLLRPCapabilities. The names of the categories are quite explicit and I won't detail all these groups except the General Device Capabilities which correspond to the data used in the controller.

The information send in the General Device Capabilities are for example the support or not of GPI/GPO and the number of ports, the manufacturer identifier, the firmware version, the maximum number of antennas to connect, the model identifier but also more technical information with the number of Air Protocols supported by the antennas or the sensitivity range received for each antenna.

In this controller we use this request to know the number of GPI/GPO ports available and the general characteristics such as the firmware version or the manufacturer identifier.

Reader Device Config Section

This section is dedicated to the functions used for the configuration, only the function to set the configuration is implemented in our controller.

SET_READER_CONFIG

This command, sent by the Client to the Reader, is used to set the reader configuration using the parameter defined by the Client. The values configured by this command are valid while the LLRP connection with the reader is open and until a new SET_READER_CONFIG message.

Example of application

> *ROReportSpec parameter defines the reporting of operation reports format and trigger for a ROSpec. ROReportSpec sent as part of SET_READER_CONFIG becomes the default ROReportSpec for the Reader. A ROReportSpec sent as part of ROSpec in the ADD_ROSPEC command overrides the default value for that ROSpec. However, in cases where there is no ROReportSpec specified in a ROSpec sent as part of ADD_ROSPEC, that particular ROSpec inherits the default ROReportSpec.*

This works same with the other parameters, like the GPI/GPO configuration, antenna gain configuration, events notification configuration and the keepalive configuration.

There is also a ResetToFactoryDefault field used to reset to the default configuration before applying the remaining parameters.

SET_READER_CONFIG_RESPONSE

This is the response by the Reader to a SET_READER_CONFIG command. If all the parameters specified in the SET_READER_CONFIG command are successfully set, then the success code is returned in the LLRPStatus parameter. If there is an error, the appropriate error code is returned in the LLRPStatus parameter.

Controller development

The following part deals about the development of the reader controller. The development was in three steps. The first one was to familiarize myself with LLRP, using all the functions.

Then the second step was about working on a basic controller with a GUI but only the basics features and finally the third one was to build a complex controller, with almost all asked features.

Introduction to LLRP : Basic controller for one reader

After the ratification of the LLRP standard a group of companies and universities started working together in order to build a robust toolkit to purpose to the public [14]. The objective of this development was to enable an accelerate LLRP development. This toolkit is available in different programming languages, such as C, C++, .NET, JAVA or Perl.

After my work on identifying the required LLRP features I started working on a basic implementation of LLRP, based on this LLRP toolkit in C language.

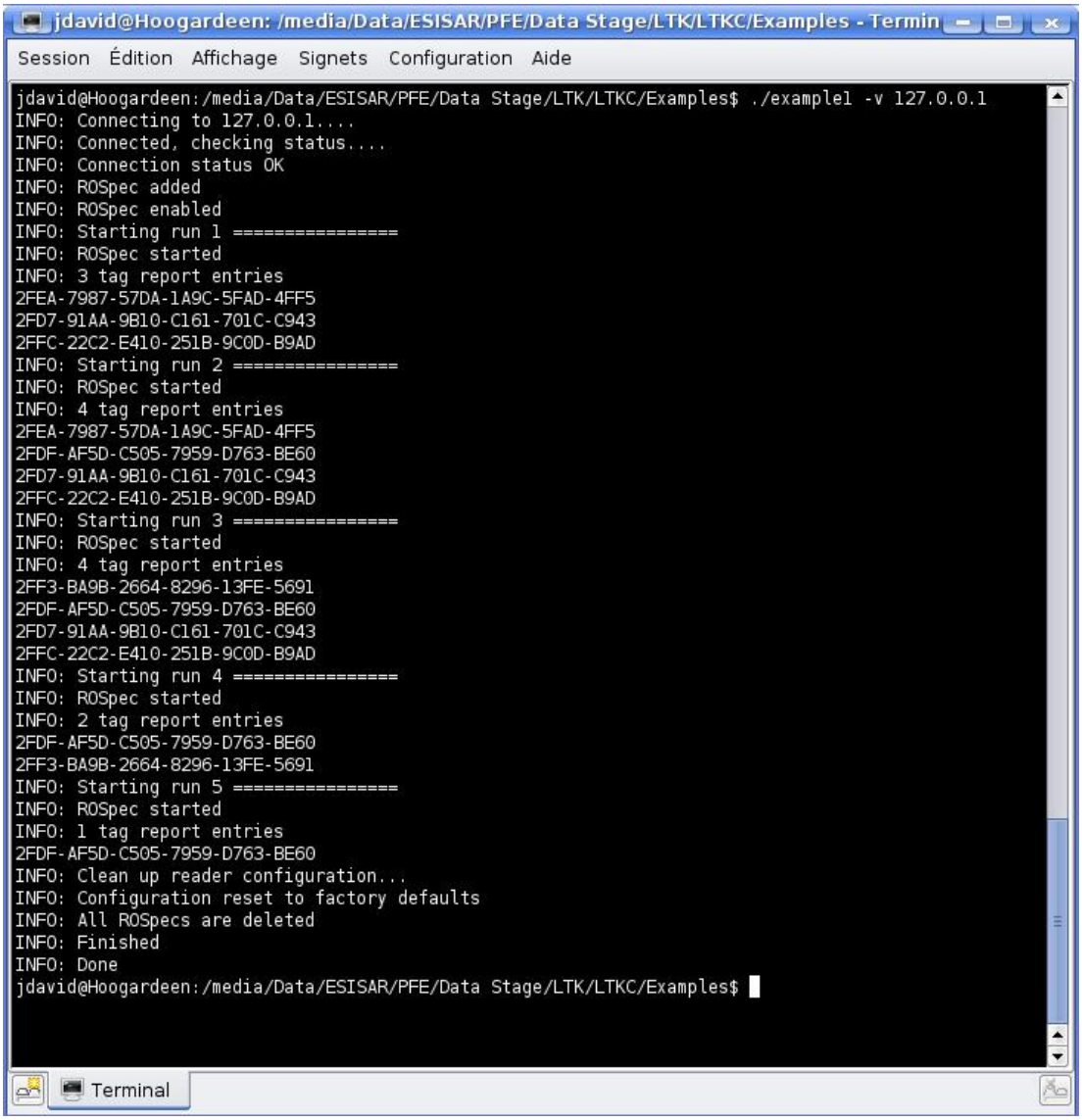


Figure 12 - Screenshot of the modified example from the toolkit

As said in introduction the aim of this controller was to play with the LLRP functionalities, to get knowledge about LLRP and the way it works. There were not precise objectives in terms of functionalities. As there was not any GUI, developing in C language was the faster way build this program.

The first step was to study all the provided examples in the toolkit. Sometimes modifying it to understand how it worked and to solve some bugs (fig 12). The development step was done with RIFIDI Emulator as simulator, the problem is that there are some LLRP implementation difference in both the simulator and the toolkit. Some parameters are required by RIFIDI and are not in the example programs or some the toolkit allows the user to start a ROSpec which has not been enabled.

This was the first part of my job on this controller, compile the examples, run them, and understand why they work or did not work and finally solve the problems.

Once I had these examples running correctly I built my own functions and a basic program with the objective to have a program able to connect to the reader and send only one or two basics commands. For example, as shown on the *fig 13*, the sequence could be:

- Connect to the reader
- Send a basic SET_READER_CONFIG message
- Close the connection

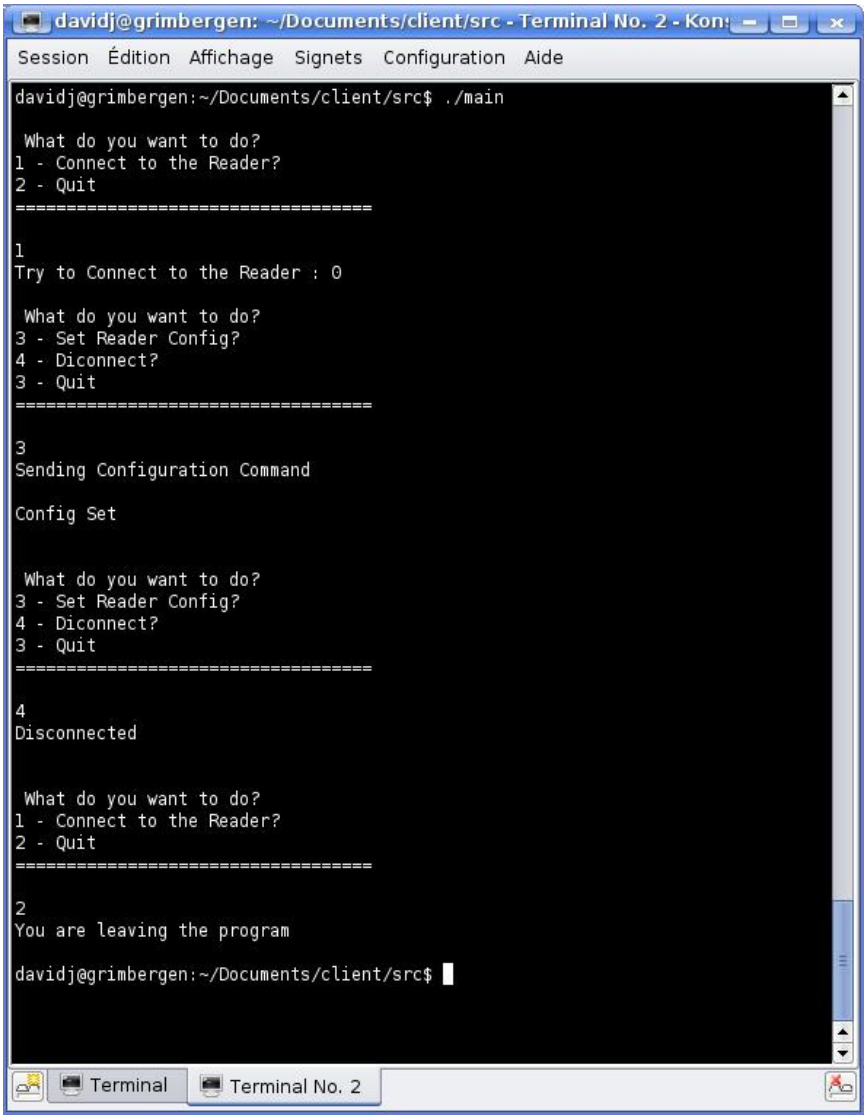


Figure 13 - Screenshot of the basic C program

Of course this basic program was only able to manage one LLRP connection at the same time.

Basic controller for two readers

The two mains improvements of the new version of the controller were:

- Support of connecting to two readers
- Addition of a graphic user interface

In order to move closer to the requirements is was indispensable to manage to connect to several readers at the same time and to add an interface able to present clearly of the features provided by the controller.

For these two reasons I selected the C#.NET language instead of C language (the Toolkit is available also in C#). I fully beneficiated of the advantage of an object

oriented programming (OOP) language in the development of a big software, using classes, objects, inheritances and all the general concepts of the OOP.

The choice of the language was mine and I choose C# because of my experience using it during my industrial project in 4th. Developing in C# allows also using Visual Studio with the WYSIWYG graphical user interface tool. It makes save a lot of time in the development of the graphical user interface.

Connecting to two readers

Note: *During the debugging step I continue working with RIFIDI Emulator but for the second reader I made tests with a real RFID reader from Impinj[15]. The appendix D shows how to set up properly RIFIDI Emulator in order to use it with the controller.*

This was the main work of this version, allowing the controller to connect to multiples readers. Instead of only one LLRP connection trough UPD there were two. The user can add the IP or the hostname of the reader and choose to connect it whenever he wants.

Then the main difficulty was to know which reader is active and especially where to send the commands and which reader is answering, or which reader is sending an operation report. Briefly the main difficulty is not to connect to the two readers, it is to manage these connections when the program is running.

Here the OOP was helpful, an object called *selected reader* was defined and was a reference to the active reader object. Every time the user changed the active reader, or an event changed the active reader this reference followed and this reference was passed in argument to the functions. This solution works only for the outgoing message, but there is still a problem to identify the senders of the incoming messages.

Concerning the incoming messages the solution is different. Actually the incoming messages are responses to the commands or operation reports.

- In the first case the response arrives just after the command has been sent, so if the incoming message corresponds we can imagine that it is the answer. We can verify that because in each ……_RESPONSE message there is a parameter with the identifier of the message that it responds to. Every time that the engine sends a message, it generates an identifier and check automatically that the response corresponds to this identifier.
- The second case is different because the incoming messages are not answers. When the user set up a reader operation he can choose to receive periodic reports, with all scanned tags. When this kind of message it is possible to identify the operation corresponding to this report. Once again a parameter field indication the ROSpecID that

correspond to the operation. So we just have to check which reader has a reader operation configured with the specified Id. The only constraint is to manage the ROSpecIDs for the two readers together and not separately, this is not really a constraint, it was easier to manage only one list of used ROSpecIDs than one for each connected reader.

Added functionalities

Regarding to the previous version the following functionalities have been added:

- Add a ROSpec and visualize the tag reports
- Get the reader capabilities and process the data received
- Set up the entirely the tag reports, the access spec reports and also the antenna parameters.

All these features are built on the same pattern, after selecting the active reader and clicking on the corresponding but a form open in order to change the parameters. The SET_READER_CONFIG form is more complex because of the important number of parameters. When closing the form the message is sent to the active reader and the answer is displayed on the main form.

This controller still needs improvements to be useful. Even with the added features the possibilities of the user are quite limited, the basic sequence is:

- Connect to the reader
- Get the reader Capabilities in order to have information about the support or not of GPI/GPO and the number of antennas
- Set the reader configuration :
    - Reader Event Notification - Enable or not different kind of event notification
    - Antennas properties – Disconnect all the antennas or set up the gain
    - Reader Operation Report – Enable or not the different kind of information about the tag in the Reader Operation reports
    - Access Report – Configure the frequency of the Access Reports
    - Keepalive Parameters – Enable the Keepalive and set the period
    - GPO/GPI port – Enable or not the GPI/GPO ports
- Add, Enable and Start a Reader Operation
- Visualize information about the scanned tags, these information are the first seen date, last seen date and the ID of the tag

For the moment the user cannot really act according to the results he has, because he doesn't really have results. This will be the goal of the next version, give more possibilities to the user according to the results he obtains.

Another point will be to expand the number of connected readers.

Complete controller for two readers or more

For this new version the improvements were:

- Connecting to multiples readers
- Support of time multiplexing
- Possibility to restart a finished RO
- Support of tag filtering
- Detection of idles tags
- Graphical interface improvements

Figure 14 - Class diagram of the re RFID reader controller

The first objective of this new version was to enable the connection to more than two readers. It has been done by changing the data structure to manage the different connections. Using advantages of the OOP a *Reader* class has been defined (fig14) and an instance of this class was created for each connected reader. Then all these objects were stored in an appropriate collection. The problem of the active reader was still present and solved with the same solution. The functionality of deleting a reader from the list of connected readers has been added too. But the reader must be disconnected before delete it.

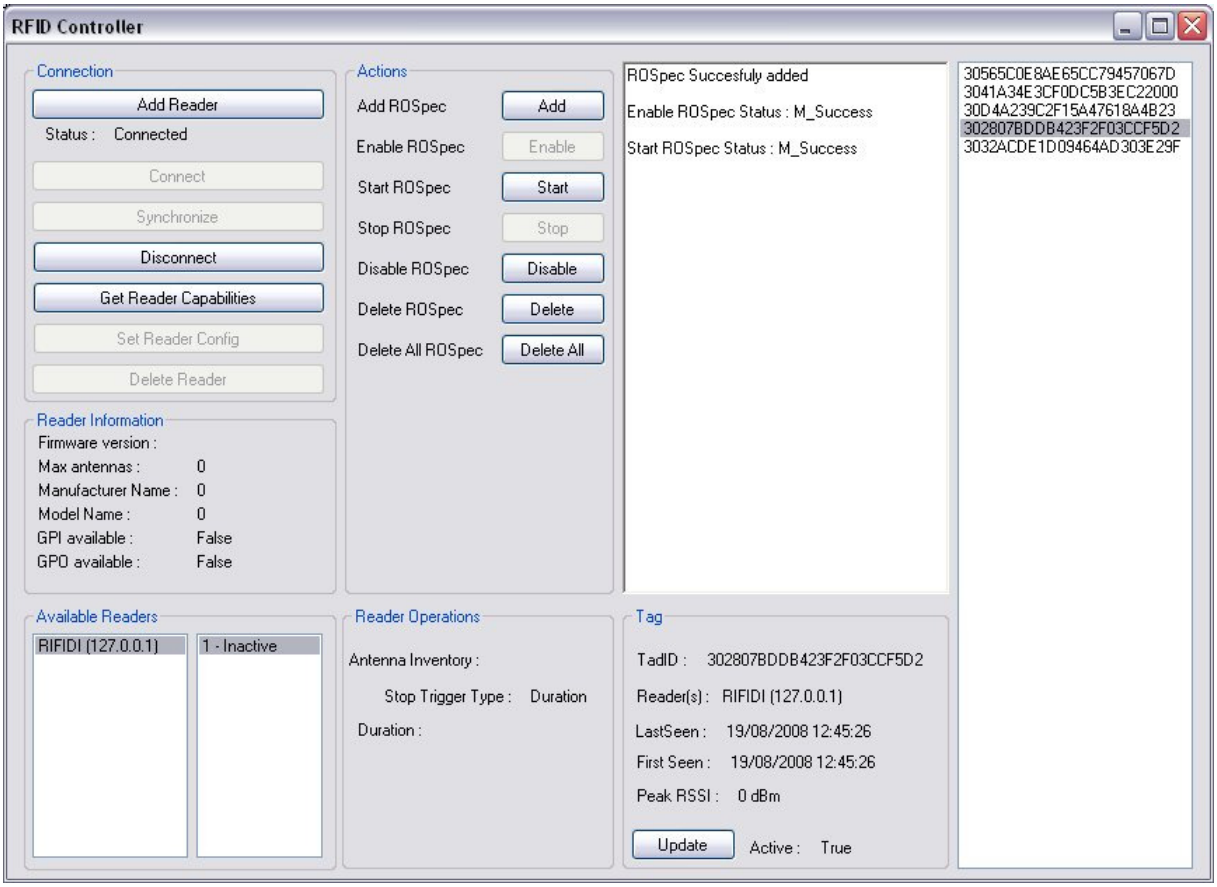


Figure 15 - Main interface

The second improvement was about updating a tag report. When a tag is selected in the tag list, information about itself appears in the tag area (fig 15). Among these information appears the reader(s) that scanned this tag. If the user clicks on the update button, an operation will be set up on the corresponding readers to update the tag status. If there is more than one reader who scans at least once this tag it is impossible to configure the same operation on these readers at the same time.

If we do the readers will scanned the same area at the same time and their signals will interfere. Here the solution of *time multiplexing* is used. The operations configured on the readers are periodic operations with an offset, depending on the number of readers to configure. The offset changes in order to avoid the readers scanning at the same time.

Another new feature is the possibility for the user to restart a finished reader operation. In the Reader Operations area (fig15) appears the information about the selected RO. If the operation corresponds to the one the user would like to add he can just start it again. The START_ROSPEC message will be sent with the ID of the corresponding RO.

The tag filtering has been implemented too. This is a feature provided by the EPC Global C1G2 protocol. It allows the reader to define a mask on one of the bank of the tag memory. The mask can have a variable length and can be applied to all the

memory banks. The figure below (fig 16) shows that it is also possible to select the trunk action, if the tag must or mustn't match to the mask.

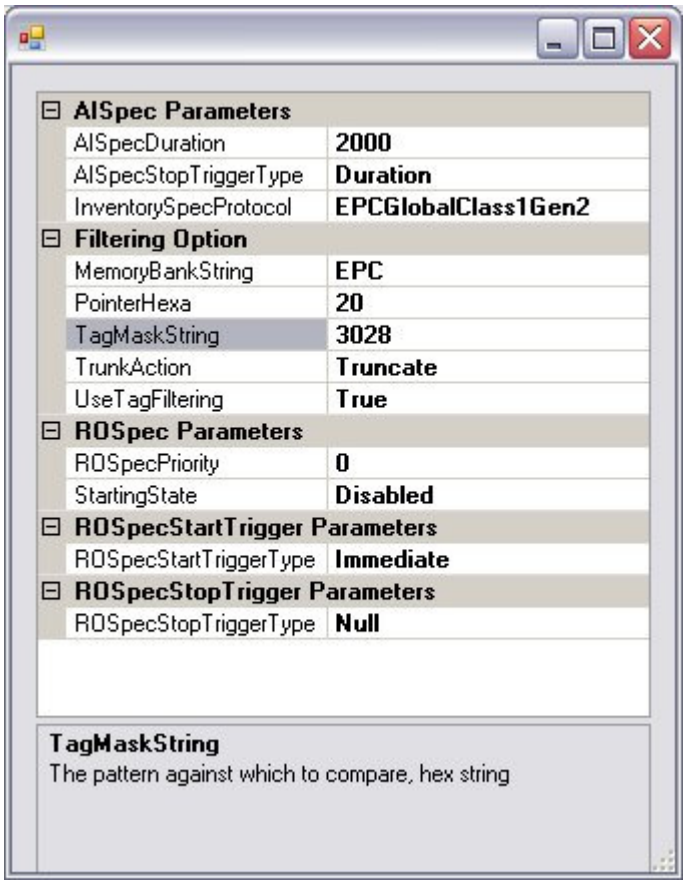


Figure 16 - Add ROSpec form with tag filtering configuration

This is an example of features offered by an Air Protocol – protocol used between the reader and the tags – used with LLRP.

The detection of idles tags is the last improvement of the last version of the controller. The problem was that the tags were displayed in the tag list all the time, once they have been scanned and with no distinction between the actives tags and the idles tags. The idles tags are those who have been scanned once by a reader and who were not present anymore after a second scan. In this new version there is an *active field* in the tag information area (fig 15) that indicates if the tag has been read during the last scan of the reader.

In order to implement this functionality, each time that a reader sends a report, the controller must check if all tags from lasts reports are still active.

The modifications of the graphical interface won't be detailed, before these modifications the interface was a *debugging interface* without any reflection about the ergonomics and the work during this last step was to create the interface shown on the figure 15. The appendix A is a user manual that explains how to use this graphical interface.

For technical purpose the programming documentation has been written and generated in html format. This document is not really dedicated to be printed, nevertheless the summary is presented in the appendix B with an extract of one class documentation.

Financial estimation

This exploitation sheet is established on the bases on :

- 5 months for the project duration
- 20$ per day for general charges of University (energy, services, cleaning…)
- 80$ per hour for tutorial management

Charges	Invest cost ($)	Total exploitation duration (m)	Project duration	Project cost ($)
Workstation PC	750	36	5m	104
RFID Kit	3500	36	5m	486
Furnish material (desk…)	300	84	5m	18
Tutorial management			20h	1600
General exploitation charges			105d	2100
Total Budget				**4308**

Conclusion

As this project was about a technology I didn't know before it, I needed a training period before really start working on it. It allows me to get a lot of knowledge about the radio frequency environment in general and the RFID environment in particular.

These information is presented in the introduction, I thought that someone who wouldn't have any knowledge about RFID would understand enough this technology to read this document.

Working on a development project allows me to improve my programming skills. After my industrial project in 4th year, which was a development project too, we had a course of Software Engineering during the 5th year. This course gave me essentials methods during a development project and applying them made me save a lot a time. I really felt the differences between my two development project in terms of methodology and my approach of the project was really different.

Paradoxically working alone on this project made me feel the importance of the work in team. All the exchanges we had during the industrial project, when we worked in a team of 3 students, were not present there. In this project, as I worked alone, there were only few discussions and it made really a difference in the way of following the project.

And of course being in Canada during this project allows me to practice and improve my English during five months. The human and cultural aspects were very present during this project too and I really learnt many things about the work in another country with another culture that the ones I knew.

Regarding to the project specification the project is not finished yet. Only the base of the RFID reader controller is built, with the essentials functions but some required features are not developed yet.

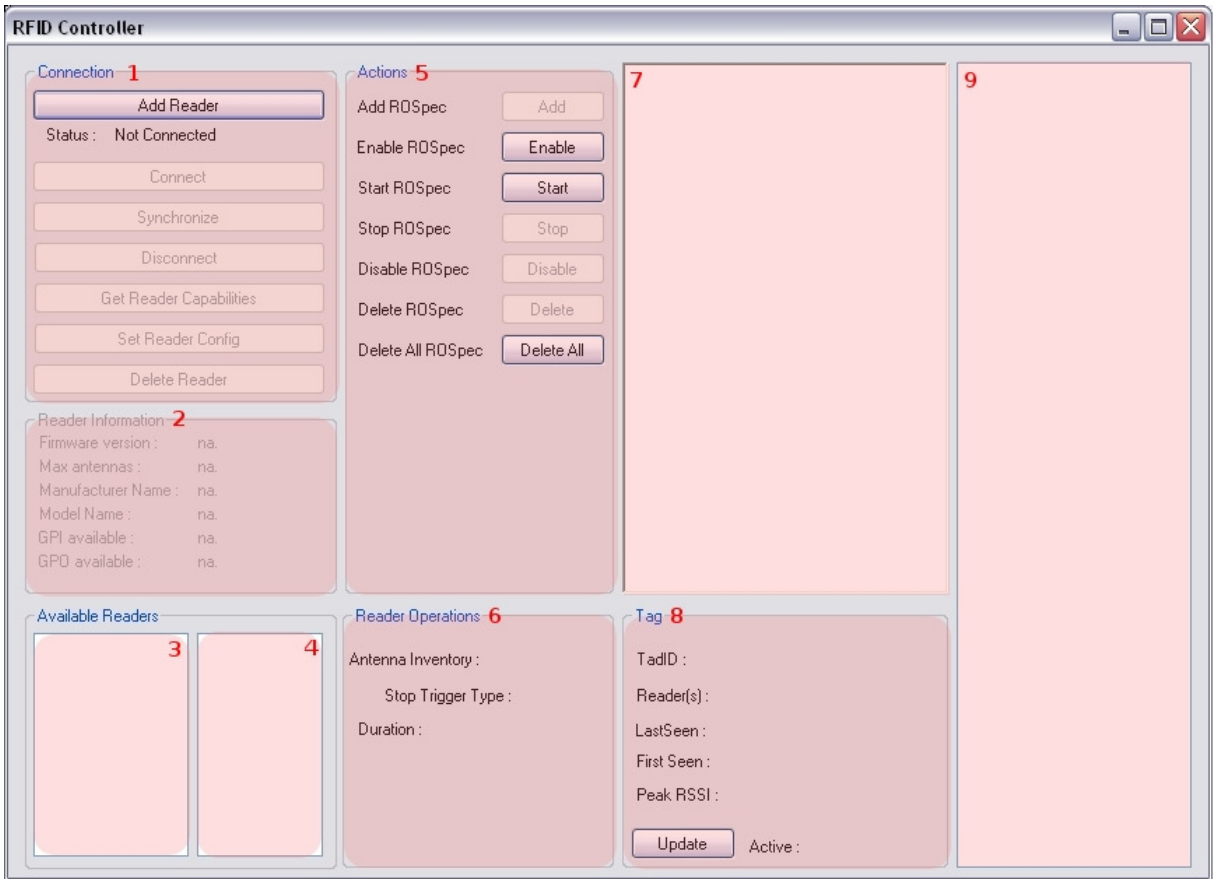
The most important thing to do now is testing the controller with a large amount of connected readers. As OPNET modeler was very long to master I didn't work with it and I had only two readers when debugging the controller. This is not enough to be sure that the controller is really scalable.

Also because of a problem of readers I did not develop the features about detecting tags read twice or more or moving tags. As the two readers I had, one simulated and I real one, did not work in the same environment, they did not read the same tags so it was impossible to detect the moving tags and the tags read twice.

Appendices

Appendix A - User Manual

Graphical User Interface Presentation



This first paragraph will present the different areas of the controller:

1. Action on the reader area, all the buttons to connect, synchronize, get capabilities, configure the reader are in this zone
2. Reader Information zone, this is the area where all the information about the reader are displayed : firmware version, number of antennas…
3. List of available readers, all the reader added by the user are displayed here. Then the user can select the active reader and make all the actions.
4. ROSpec list, all the created ROSpec on the selected reader are displayed here with its id and its status.
5. ROSpec actions zones, all the buttons to create, enable, start, stop, disable and delete ROSpec are in this zone.

6. Reader Operation information zone, characteristic about the selected ROSpec are displayed in this area.
7. Reader Answers area, all incoming messages from the readers are displayed here
8. Tag information area, the information about the tags are displayed here, first seen date, last seen date, ID, status…
9. Tag zone, all scanned tag are displayed in this area

Setup a Reader Operation

The following part explains how to connect a reader and configure a basic reader operation.

To connect the reader you have to first add the reader in the reader list. To do this, simply click on the *Add Reader* button and fill the blanks:

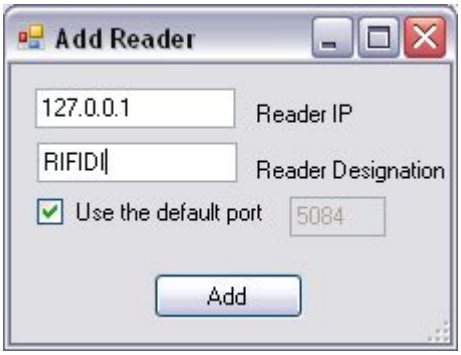


Figure 17 - Form to add a reader in the controller

The field Designation corresponds to the name of the reader that will be displayed in the reader list. The port field designs the LLRP port used by the reader, the default port is 5084.

After clicking of the *Add* button, the reader is now displayed in the reader list :

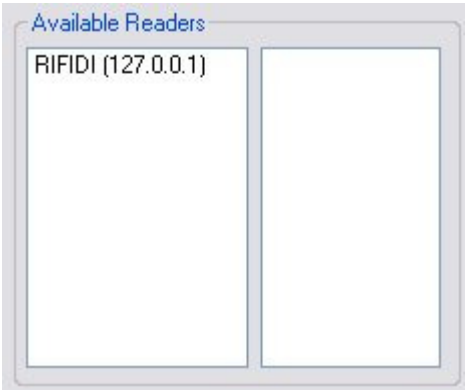


Figure 18 - New reader displayed in the reader list

The following step is now to connect to the reader by selecting the reader and clicking on the *Connect* button. After doing that new actions are available :

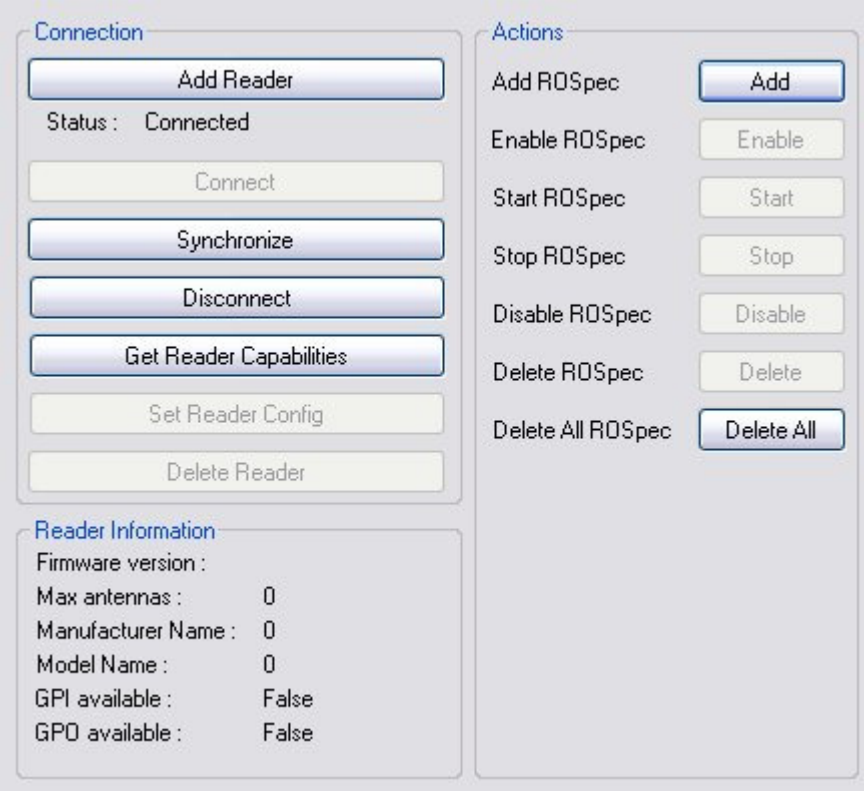


Figure 19 - New features available when the reader is connected

Click on the *Get Reader Capabilities* in order to get information about the reader and then on the *Add* button in the ROSpec action zone. A new window opens, where you will set up the ROSpec :

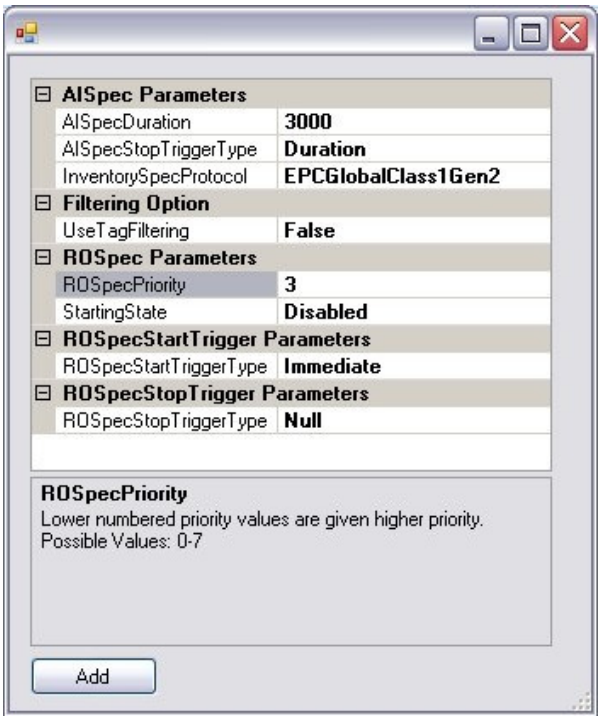


Figure 20 - Form to set up a new ROSpec

The configuration shown on the figure above will set up a ROSpec with an Antenna Inventory Spec which will last 3sec, without tag filtering and with a priority of 3. After clicking on the *Add* button this window closes, the ADD_ROSPEC message is send, the ROSpec id added in the list and the *Enable* button is available :

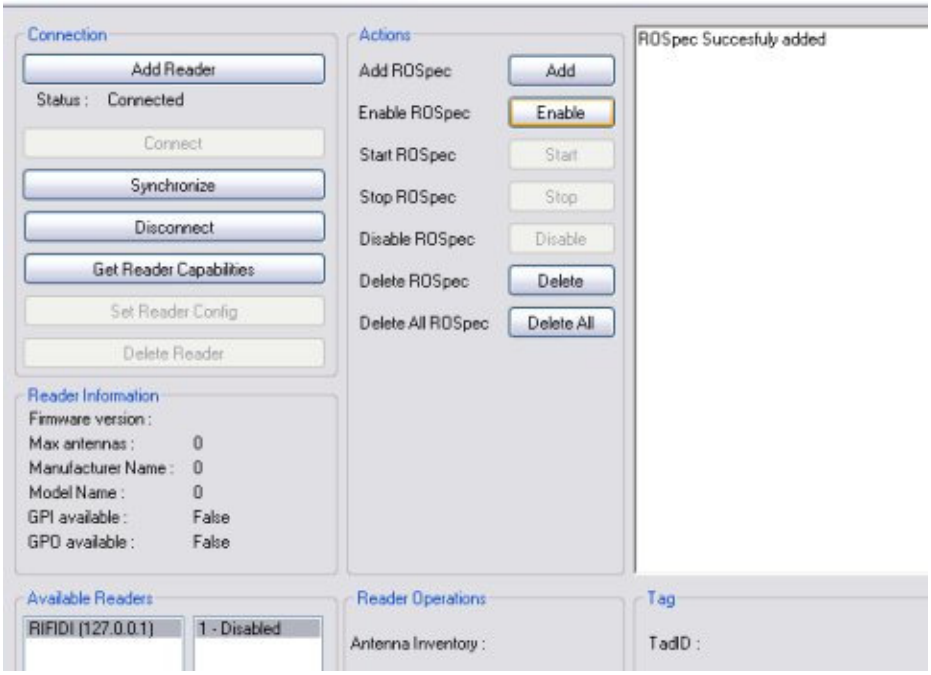


Figure 21 - New ROSpec Added

Enable the ROSpec by clicking on the *Enable* button, then start it by clicking on the *Start* button. After 3 seconds, the time that we configured and we created the ROSpec, the controller receives a report from the reader and all the tags appear in the tag list :

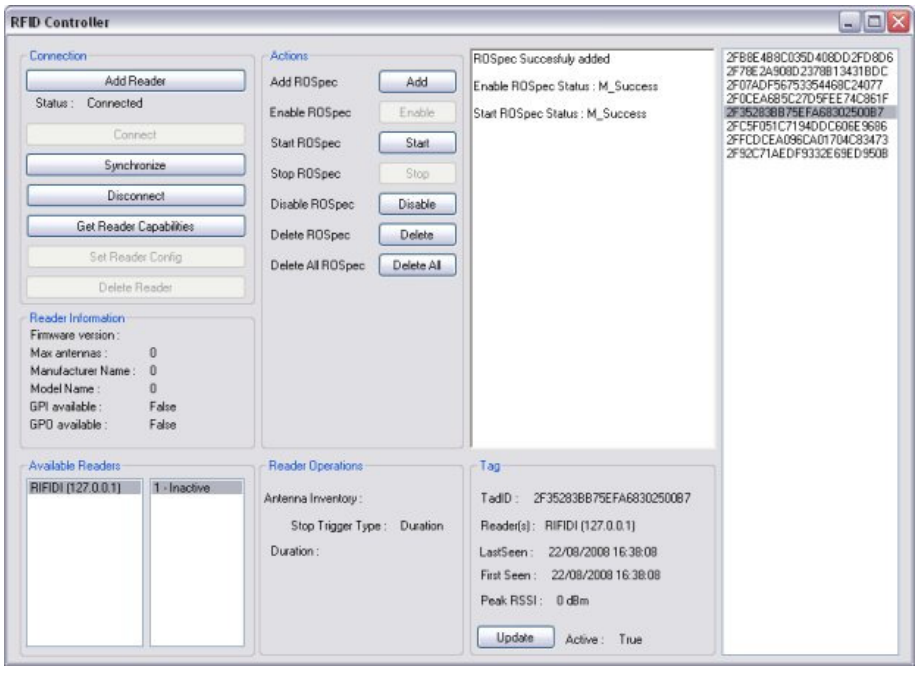


Figure 22 - Tags added in the tag list

You can now select a tag and read the information about it.

Setup the tag filtering

The following part explains how to setup the tag filtering. We will configure a new ROSpec that will scan only some tags.

We have first to create another a ROSpec, but now enabling the tag filtering :

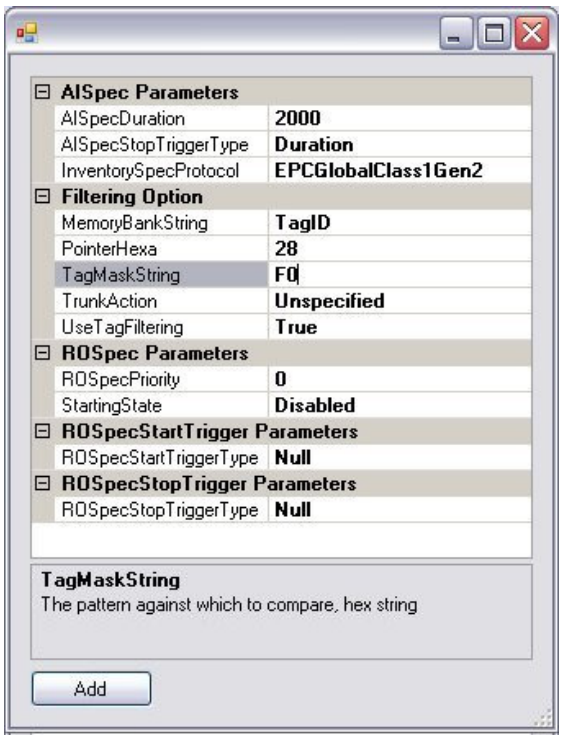


Figure 23 - Configuring a ROSpec with the tag filtering

We will now explain the parameters that we set :

- MemoryBank, we select now the bank of memory on which apply the mask. Here *TagID* is selected, that means that the tagID will have to match with the mask.
- Pointer, this is the offset to apply on the mask. As there are two reserved words of 16bits in the *TagID bank* before the tagID the default value is 20. The value of 28 indicates that we will apply the mask on the second and third words of the tagID
- TagMask, this is the value of the mask in an hexa format.
- TrunkAction, this is the truncate action, if it is unspecified the reader decides what truncate action to take. If it is *truncate* the reader report only the portion of the ID following the mask.

Now click on the *Add* button to add the ROSpec in the ROSpec list, then select it, enable it and start it.

The reader will report only the tags starting with : *XF0*

Appendix B – Source Code documentation

Collapse All Code: C#

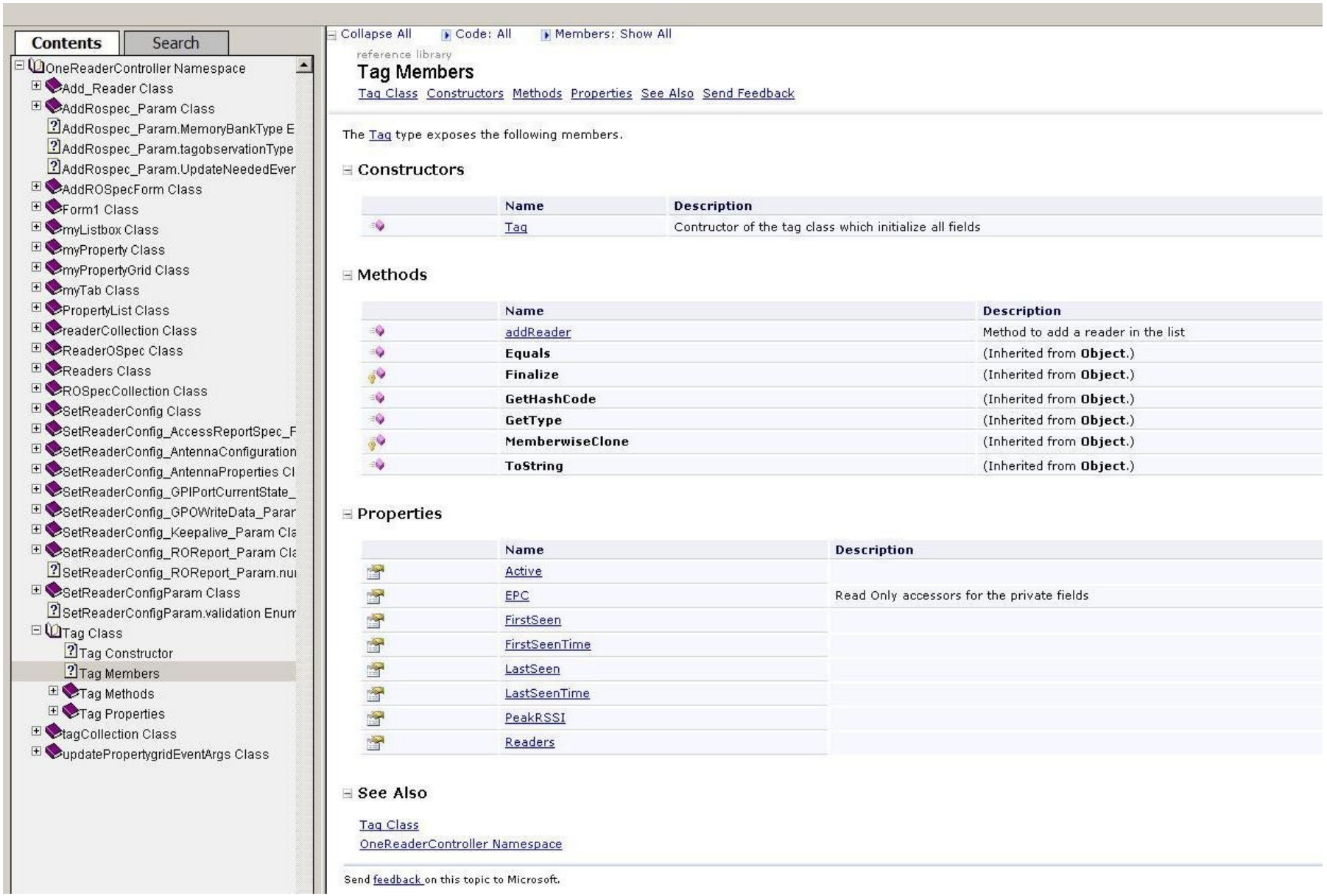
reference library

OneReaderController Namespace

Send Feedback

Classes

Class	Description
Add_Reader	
AddRospec_Param	This class is used to define all the parameters for the add RoSpec Form This class must extend ICustamClass to be compatible with the MyPropertyGrid control
AddROSpecForm	This form define the GUI to parameter a new ROSpec
Form1	This is the main window of the program All features are availables from this main window, such as add a reader, configure it and process the data received from the readers
myListbox	
myProperty	
myPropertyGrid	
myTab	
PropertyList	Class used to set up the dynamic PropertiesListbox in the AddROSpec form
readerCollection	This is a class extended from ArrayList in order to add a specific search method This class is used to store the readers objects
ReaderOSpec	Class used to represent the ReaderOperation Spec
Readers	This class reprents the readers One instance of Readers is created for each declared reader in the GUI The dialog between the controller and the connected readers are down via this class
ROSpecCollection	Class extented from ArrayList to add some specific method This class is used to store all ROSpec
SetReaderConfig	
SetReaderConfig_AccessReportSpec_Param	This class is used to define all parameters for the Set reader config message
SetReaderConfig_AntennaConfiguration_Param	This class is used to define all parameters for the Set reader config message
SetReaderConfig_AntennaProperties	This class is used to define all parameters for the Set reader config message
SetReaderConfig_GPIPortCurrentState_Param	This class is used to define all parameters for the Set reader config message
SetReaderConfig_GPOWriteData_Param	This class is used to define all parameters for the Set reader config message
SetReaderConfig_Keepalive_Param	This class is used to define all parameters for the Set reader config message
SetReaderConfig_ROReport_Param	This class is used to define all parameters for the Set reader config message
SetReaderConfigParam	This class is used to define all parameters for the Set reader config message
Tag	Class used to represent the tags Each read tag corresponds to an instance of this Tag Class
tagCollection	This is a class extented from ArrayList in order to add a specific research method This class id used to store all the tag object
updatePropertygridEventArgs	This class is used to declare the event to update the dynamic propertie grid in the add rospec form


Contents
Search
OneReaderController Namespace
Add_Reader Class
AddRospec_Param Class
AddRospec_Param.MemoryBankType E
AddRospec_Param.tagobservationType
AddRospec_Param.UpdateNeededEver
AddROSpecForm Class
Form1 Class
myListbox Class
myProperty Class
myPropertyGrid Class
myTab Class
PropertyList Class
readerCollection Class
ReaderOSpec Class
Readers Class
ROSpecCollection Class
SetReaderConfig Class
SetReaderConfig_AccessReportSpec_F
SetReaderConfig_AntennaConfiguration
SetReaderConfig_AntennaProperties Cl
SetReaderConfig_GPIPortCurrentState_
SetReaderConfig_GPOWriteData_Parar
SetReaderConfig_Keepalive_Param Cla
SetReaderConfig_ROReport_Param Cla
SetReaderConfig_ROReport_Param.nui
SetReaderConfigParam Class
SetReaderConfigParam.validation Enum
Tag Class
Tag Constructor
Tag Members
Tag Methods
Tag Properties
tagCollection Class
updatePropertygridEventArgs Class
Collapse All
Code: All
Members: Show All
reference library
Tag Members
Tag Class Constructors Methods Properties See Also Send Feedback
The Tag type exposes the following members.
Constructors
Name
Description
Tag
Contructor of the tag class which initialize all fields
Methods
Name
Description
addReader
Method to add a reader in the list
Equals
(Inherited from Object.)
Finalize
(Inherited from Object.)
GetHashCode
(Inherited from Object.)
GetType
(Inherited from Object.)
MemberwiseClone
(Inherited from Object.)
ToString
(Inherited from Object.)
Properties
Name
Description
Active
EPC
Read Only accessors for the private fields
FirstSeen
FirstSeenTime
LastSeen
LastSeenTime
PeakRSSI
Readers
See Also
Tag Class
OneReaderController Namespace
Send feedback on this topic to Microsoft.

Appendix C – Synthesis in french/Synthèse en français

Ce document traite de mon projet de fin d'étude effectué au premier semestre 2008 dans un laboratoire de l'université d'Ottawa au Canada, plus particulièrement dans le département ingénierie : Ecole de Technologie de l'Information et d'Ingénierie (EITI). Le sujet de ce projet étant *« Design d'un contrôleur de lecteur RFID »*, ce rapport commencera par une rapide introduction sur les concepts clé de la technologie RFID. Ensuite je présenterai le protocole LLRP qui a été étudié puis implémenté dans ce projet, avant de développer plus en détail le travail effectué pendant ce projet.

Introduction

La RFID est une technologie sans fil qui utilise des signaux de radio fréquence. C'est une technologie émergente qui est de plus en plus utilisé dans l'industrie, pour des applications diverses allant du suivi sur une chaîne de production à l'identification des produits dans un entrepôt. Dernièrement cette technologie à été intégrée dans certains passeports pour une lecture rapide des informations de bases.

La clé de cette technologie repose sur le dialogue entre les deux entités clé, d'un coté le lecteur et de l'autre l'étiquette (ou tag). Le tag est un élément doté d'un micro circuit intégré avec sa mémoire et d'une antenne. Le signal électromagnétique émit par le lecteur via son antenne suffit à alimenter le tag qui peut ensuite décoder le signal reçut et, le cas échant y répondre. Ce modèle de communication implique donc que le lecteur initialise l'échange à chaque fois puisque le tag n'est alimenté qu'une fois le signal reçu.

Comme dans toutes les technologies sans fil il existe un problème de collision lorsque deux, ou plus, entités essayent d'accéder au canal. Il existe différentes sorte de collision, deux lecteurs qui scannent la même zone ou encore deux tags scannés qui répondent au même moment. Nous nous intéresserons plus particulièrement aux collisions mettant en cause deux lecteurs (reader to reader collision) et verrons la solution mise en place pour les éviter.

Le paragraphe suivant va présenter un peu plus en détail les deux éléments important de la technologie RFID, les lecteurs et les tags.

Un lecteur RFID est composé de plusieurs parties :

- Un module RF
- Une ou plusieurs antennes

Le module RF est le cerveau du lecteur, c'est dans cette partie que sont générés les signaux à envoyer et où sont traitées les réponses des tags. C'est sur ce module maître que sont reliées les interfaces Ethernet et/ou série. En effet les lecteurs RFID disposent d'une connexion réseau permettant à un utilisateur ou un logiciel de se

connecter au module RF pour toutes sortes d'opération spécifiques. L'autre partie du lecteur est l'antenne. Un lecteur a au moins une antenne, qui lui permet d'effectuer le dialogue avec les tags, mais il peut en avoir jusqu'à quatre et ainsi avoir une zone de lecture très étendue. Le fait d'avoir plusieurs antennes permet à un lecteur d'avoir différents types d'opération effectuée en même temps sur des zones différentes.

Le deuxième interlocuteur lors d'un « dialogue RFID » est le tag. Ce petit circuit composé, pour les plus simples, d'une antenne et d'un circuit intégré. Il existe des tags appelé actifs, qui ont en plus une batterie, ce qui leur permet d'émettre leurs informations en continue, sans nécessité de recevoir au préalable une requête de la part du lecteur. La puissance présente dans le tag étant beaucoup plus importe ces tags peuvent être lus depuis des distances bien plus importantes. En contrepartie ils coutent beaucoup plus cher à développer et sont beaucoup plus gros que leurs homologues *passifs*.

Le paragraphe suivant va présenter le protocole LLRP (Low Level Reader Protocole). Ratifié en avril 2007 par l'EPC Global en tant que standard, ce protocole sert à standardiser l'interface réseau des lecteurs. Attendu depuis plusieurs années, ce protocole est vu par beaucoup d'industriels comme l'élément manquant pour promouvoir la technologie RFID à un très haut niveau d'utilisation.

L'unité de données de LLRP est le message, ces messages sont transmis du Client vers le Lecteur ou du Lecteur vers le Client. Dans LLRP est appelé Client toute entité qui dialogue avec un lecteur RFID. Ce client peut être logiciel mais il peut être matériel aussi, c'est pour ça que cette désignation a été choisit, car elle ne suggère pas d'idée quand a l'état du Client.

En utilisant des messages LLRP, le Client met à jour l'état du lecteur, cela inclut les paramètres de configuration du lecteur, les structures de données crées dynamiquement (ROSpecs…) et éventuellement d'autre donnée propriétaires. C'est pourquoi LLRP nécessite des accusés de réception pour les échanges entre lecteur et client.

Les *Reader Operation* sont un des types d'opération qu'un client peut paramétrer sur un lecteur. C'est l'opération qui a été la plus largement utilisée dans le développement du contrôler c'est pourquoi le paragraphe suivant ne traitera que de ce type d'opération.

Une *Reader Operation* définit les paramètres pour un traitement à exécuter par le lecteur. Elle contient au moins une spécification, que ce soit une spécification pour une opération de type inventaire (AISpec) ou une spécification pour une opération de type radio fréquence (RFSurveySpec).

Le principe d'une opération de type inventaire est que le lecteur va interroger tous les tags présents dans sa zone. Pendant une opération de type radio fréquence le lecteur

effectue des mesures de puissance du signal au niveau des antennes et cela dans des gammes de fréquences différentes.

Développement du projet

La partie suivante aborde le développement du projet, ce projet s'est déroulé en trois phases :

- Une étude comparative des différents simulateurs de lecteurs RFID existants
- Une étude sur le protocole LLRP
- Développement du contrôleur de lecteur RFID

Etude comparative sur les simulateurs RFID

L'objectif du projet étant de construire un contrôleur de lecteurs RFID capable de gérer jusqu'à 100 lecteurs il est bien entendu impossible dans un premier temps de faire fonctionner une centaine de lecteurs pour le développement. La solution est donc d'utiliser un logiciel de simulation de lecteurs RFID.

Il existe un certains nombres de ces logiciels parmi lesquels :

- RIFIDI
- Tag Centric
- RFID Anywhere
- OPNET Modeler

La première phase du projet a donc été d'identifier le simulateur adéquat pour ce projet. Il a fallu donc définir les critères de sélection du simulateur et leur importance. En ayant en parallèle le cahier des charges voici la liste de critères sélectionnés :

- Simulation de lecteurs existants
- Simulation de la communication entre le lecteur et le tag
- Support des protocoles entre le lecteur et les tags
    - Support de EPC Global Class 1 Gen 2
- Support de LLRP
- Limite du nombre de lecteur simulables

La simulation du dialogue entre les tags et les lecteurs est nécessaire parce que certaines informations importantes sont échangées pendant cette communication. Par exemple le filtrage des tags est configuré sur le lecteur, puis le lecteur filtre les tags en utilisant EPC Global C1G2. Comme la fonctionnalité de filtrage des tags est nécessaire, le support du protocole EPC Global C1G2 devient tout aussi nécessaire.

Une fois les critères définie il a fallu utiliser chacun des logiciels de la liste afin d'étudier leur position vis-à-vis des critères de sélection. L'étude des logiciels open

source (RIFIDI, tag centric) a pu être faite assez rapidement du fait de leur disponibilité immédiate. Pour les logiciels commerciaux (OPNET, RFID Anywhere) il a fallu attendre d'avoir reçu l'agrément pour la licence de recherche universitaire pour avoir une idée plus précise de leurs capacités.

Les résultats de ces études sont récapitulés dans le tableau ci-dessous :

Critère / Logiciel	RIFIDI Emulator	Tag Centric	RFID Anywhere	Opnet Modeler
Lecteurs réels	Yes	No	Yes	Yes
Communication entre le lecteur et le tag	Yes	Yes	Na	Yes
Protocole entre le lecteur et le tag	Yes	No	Yes	Yes
EPC Global C1G2	Yes	No	No	Yes
LLRP	Yes	No	Yes	Yes
Limite du nombre de lecteur	6	3	Na	Au moins 25

Deux logiciels correspondent aux critères énoncés, RIFIDI Emulator et OPNET Modeler. La différence entre ces deux logiciels se fait sur le nombre maximal de lecteurs simulables. Si OPNET peut largement atteindre la centaine de lecteurs RIFIDI est nettement plus limité sur ce point là. C'est pourquoi pour la phase de test de la mise à l'échelle puis de production le simulateur utilisé sera OPNET Modeler. Cependant étant donné que OPNET Modeler est un logiciel très complexe a prendre en mains, nous utiliserons RIFIDI Emulator pendant la phase de développement du contrôleur. En effet pendant cette phase du projet seulement un ou deux lecteurs seront nécessaires pour développer et tester les premières fonctionnalités.

Etude sur le protocole LLRP

La deuxième partie du projet a été consacrée à l'étude du protocole LLRP, de ses possibilités et de ses différentes fonctions. Les fonctions de LLRP sont regroupées dans trois sections :

- Reader Operation Section
- Reader Device Capabilities Section
- Reader Device Config Section

Reader Operation Section

La première section regroupe les fonction relatives aux *Reader Operation*. On trouve par exemple des fonctions pour créer une ROSPEC, l'activer puis la démarrer. Dans l'autre sens il existe aussi des fonctions pour stopper une ROSPEC, la désactiver et la supprimer.

Chacune de ces fonctions est bâtie sur le même modèle, c'est-à-dire qu'à chaque fois il est indiqué en paramètre du message un identifiant de ROSPEC. Cet identifiant unique pour chaque opération est généré par le client lors de la création de l'operation (ADD_ROSPEC). L'action correspondante au message est ensuite effectuée sur la ROSPEC ayant l'id passé en paramètre.

La création d'une opération est le seul message un peu différent des autres. C'est la seule de ces fonctions qui comporte d'autre paramètres que l'identifiant. Comme présenté dans l'introduction sur LLRP, une ROSPEC définit un certain nombre de AISPEC ou de RFSurveySpec. Les paramètre de ces SPEC sont donc présents dans le message de création ainsi que les paramètres de rapport. Le client, via le message ADD_ROSPEC, peut choisir le type d'information qu'il souhaite dans les rapports d'opérations, il peut activer ou pas le rapport des dates de scan, l'affichage de l'identifiant de l'antenne ayant scanné le tag ou d'autre informations de ce genre.

Les fonctions présentes dans la section Reader Operation sont :

- ADD_ROSPEC
- ADD_ROSPEC_RESPONSE
- DELETE_ROSPEC
- DELETE_ROSPEC_RESPONSE
- START_ROSPEC
- START_ROSPEC_RESPONSE
- STOP_ROSPEC
- STOP_ROSPEC_RESPONSE
- ENABLE_ROSPEC
- ENABLE_ROSPEC_RESPONSE
- GET_ROSPECS
- GET_ROSPECS_RESPONSE

Les messages …_RESPONSE correspondent aux messages renvoyé par le lecteur et font office d'accusé de réception, ils contient le code de status de l'opération : succès ou erreur.

Reader Device Capabilities Section

Cette deuxième section comporte beaucoup moins de fonction que la précédente puisque seulement une fonction et sa réponse correspondante sont disponibles.

Cette fonction est une requête envoyée au lecteur, elle permet au client de demander au lecteur qui lui renvoie les informations relatives a ses capacités. Les informations présentes dans le message GET_READER_CAPABILITIES peuvent être par exemple :

- Nombre d'antennes connectées
- Support de GPO/GPI
- Identifiant du fabriquant du lecteur
- Version du Firmware

Ce sont au total une vingtaine de champs différents que le lecteur renvoie.

Reader Device Config Section

Là encore cette section est assez courte puisque une seule fonction et sa réponse sont disponibles. Cette commande permet de reconfigurer le lecteur avec des nouveaux paramètres. Le message contient une liste de huit différents types de paramètres, ces types peuvent être :

- Configuration des antennes
- Configuration des GPO/GPI
- Configuration des messages de keepalive

Le lecteur répond ensuite avec un message SET_READER_CONFIG_RESPONSE contenant le statut de la commande.

Il faut savoir que les fonctions présenté dans cette partie sont les fonctions qui ont été implémentés dans le contrôleur. Il existe d'autres fonctions qui n'ont pas été désignées comme étant indispensable pour le fonctionnement du contrôleur.

Développement du contrôleur de lecteur RFID

La partie suivante détaille le développement logiciel du contrôleur de lecteurs RFID. Ce développement s'est déroulé en trois phases :

- Une première phase d'écriture de programme C utilisant les bibliothèques de fonction de LLRP
- Une deuxième phase de construction d'un logiciel avec interface graphique en C#.NET
- Une troisième phase d'amélioration du précédent logiciel avec intégration de fonctionnalités plus complexes

Les deux premières phases n'ayant pas produit de résultats concrets pour la fin du projet elles ne seront pas détaillées dans cette synthèse. Pour plus d'informations concernant les deux premières versions du contrôleur, elles sont détaillées dans le corps du rapport.

Les paragraphes suivant sont donc consacrés au développement de la dernière version du contrôleur de lecteurs RFID. Cette version est celle qui se rapproche le plus des objectifs définis en début de projet, la majorité des fonctionnalités ont été implémentées et l'intégration des fonctionnalités manquantes a été facilité.

Le développement

Le développement du contrôleur s'est fait à l'aide de bibliothèques provenant d'un toolkit fournit par un consortium d'industriel voulant promouvoir LLRP. Afin l'adoption de ce nouveau standard un groupe d'industriels s'est regroupé pour développer des outils accélérant l'implémentation de ce protocole. Ainsi ils ont mit a disposition sur leur site internet [14] une boite à outils, disponible dans plusieurs langages de programmation, facilitant le développement d'applications supportant le protocole LLRP.

Cette boite à outils étant disponible, entre autre, dans le langage de programmation C#.NET, elle a été utilisé pour construire ce contrôleur.

Les versions précédentes du contrôleur ayant permis disposer d'une base de logiciel capable de se connecter à un lecteur pour de lui envoyer des requêtes basique, l'objectif de cette version était de rajouter des briques fonctionnelles.

La première chose à faire fût de s'assurer que le contrôleur pouvait bien se connecter à deux lecteurs ou plus. Pour faire ce test le logiciel RIFIDI Emulator [10] ainsi que le lecteur Impinj Speedway [15] ont été utilisé. Les différents scénarios ont été essayés : deux lecteurs simulés puis le troisième réels, trois lecteurs simulés en même temps, des scénarios avec des déconnections successives des lecteurs… tout ça pour s'assurer que l'exigence principale de ce contrôleur était bien respecté.

L'envoi de message à plusieurs lecteurs n'est pas plus complexe que l'envoi a un seul. Etant donné que les messages ne s'adressent qu'à un seul destinataire, le message est envoyé au lecteur dit « actif » dans le logiciel. Une liste des lecteurs étant affiché, le lecteur sélectionné est celui actif, celui avec lequel contrôleur va dialoguer.

Le problème principal d'avoir plusieurs lecteurs connecté intervient lors de la réception de rapport d'opération. En effet comme les opérations peuvent avoir des durées variables, parfois non connues du contrôleur (le client peut programmer une opération jusqu'à avoir scanné un nombre n de tags différents), il est impossible de se baser sur la chronologie des messages pour savoir qui est l'émetteur d'un rapport.

La solution qui a été retenue est de repérer l'identifiant de l'opération correspondante puis de chercher quel est le lecteur sur lequel a été configuré l'opération avec cet identifiant. Cette solution implique d'avoir une gestion des identifiants d'opération qui soit commune a tous les lecteurs connectés, afin d'assurer que deux opérations sur deux lecteurs différents n'ait pas le même ID, bien que ce serait correct au sens de LLRP.

Un autre souci majeur lors de la phase de débogage du logiciel a été les différences d'implémentation de LLRP entre RIFIDI Emulator et le standard. RIFIDI a été utilisé pour débugger en raison de son interface graphique, en effet comme il s'agissait d'un lecteur simulé sur un ordinateur, tous les messages reçus ou envoyés étaient affichés sur la console du logiciel, ceci facilitant le débogage. Néanmoins certaines fonctionnalités n'étaient pas supportées par RIFIDI telle que la prise en compte du Peak RSSI, le fitrage de tag ou encore le fait de pouvoir supprimer toute les opérations en spécifiant un id à 0 dans les paramètres.

Les fonctionnalités

La liste des fonctionnalités principales présentes dans cette version du contrôleur est la suivante :

- Connexion à un nombre illimité de lecteurs
- Requête et traitement des capacités des lecteurs
- Configuration des lecteurs
- Synchronisation du contrôleur avec les lecteurs
- Configuration d'opérations sur le lecteur
- Configuration du filtrage de tags lors d'opération de scan
- Traitement des rapports d'opérations des lecteurs
- Mise à jour de l'état des tags

La synchronisation du contrôleur avec les lecteurs consiste, lors de la connexion avec un nouveau lecteur à lui envoyer une requête pour qu'il renvoie le nombre et l'état de ses opérations configurées. Ainsi le contrôleur peut prend en compte ces nouvelles informations au moment de créer une nouvelle opération sur ce lecteur.

Le filtrage de tag lors d'opération de tag est une fonctionnalité offerte par le protocole EPC Global C1G2. Lors de la création d'une opération, l'utilisateur peut spécifier un masque et la zone de la mémoire du tag sur laquelle appliquer ce masque. Ensuite, le lecteur ne renverra de rapport d'opération que sur les tags qui correspondent a ce masque sur la zone de mémoire définie.

Chaque fois qu'un rapport d'opération de scan est reçu par le contrôleur, celui-ci met à jour l'état de chaque tag, que ce soit un tag nouvellement détecté ou un ancien. Dans le cas des anciens tags la tâche du contrôleur est de détecter ceux qui ne sont plus actif, c'est-à-dire de détecter les tags qui ont été lus une première fois par un lecteur et qui, lors d'une nouvelle opération de ce lecteur, ne sont plus scannés. La distinction est ainsi faite entre les tags actifs et les tags dit inactifs.

Appendix D - RIFIDI Emulator documentation

This appendix is a tutorial about how to set up correctly RFID Emulator in order to have a simulated RFID reader supporting LLRP.

The first thing to do is to create the reader to simulate. To do that you must click on the *Add Reader* button (see figure above).

Figure 24 - Add a reader in RIFIDI Emulator

A new window now opens, where you can choose the type of reader you want, select *LLRP Reader* as on the following figure and then click *Next*.

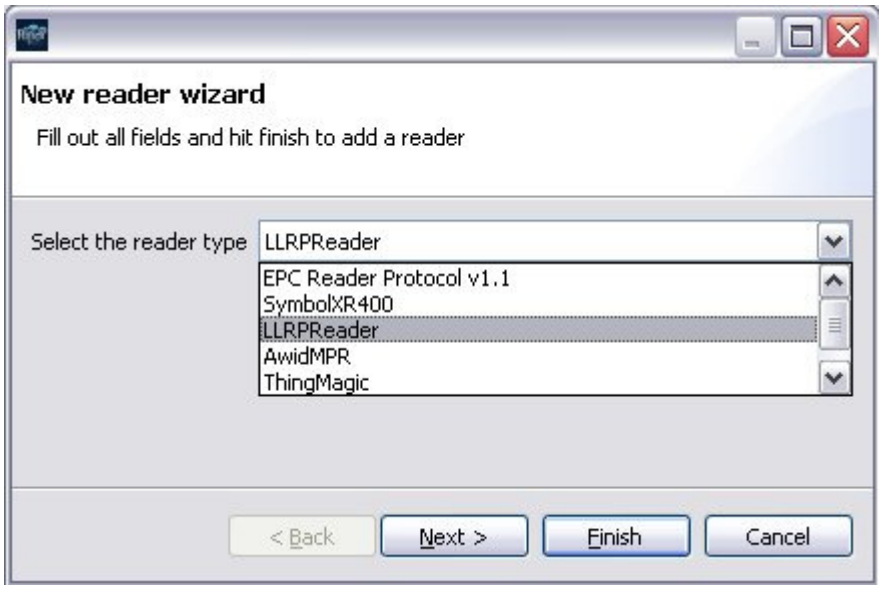


Figure 25 - Select LLRP Reader

The following window asks for the reader's parameters. You can modify the reader's name, the number of antenna and enable or not the GPI/GPO but it is strongly recommended to let the default settings for the administration IP and the LLRP IP, as on the figure above.

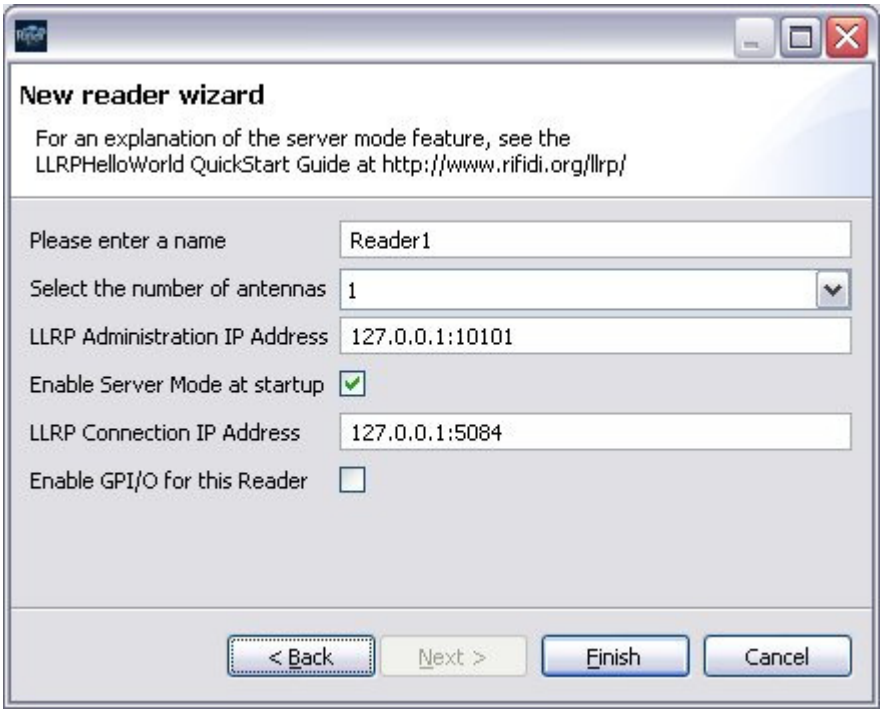


Figure 26 - Parameters for the new reader

When you click on the *Finish* button this window close and a new tab appears on the main window :

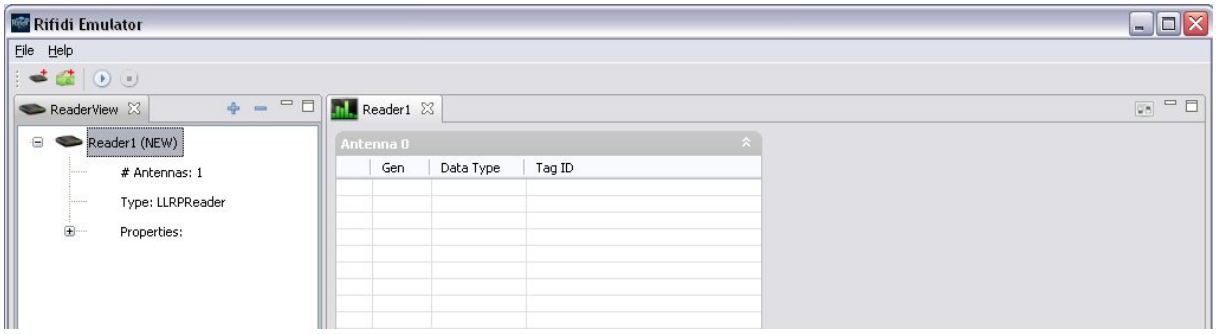


Figure 27 - New reader created

The next step is to create tags and drag them in the reading zone of our new reader. Creating tags is almost as easy as create the reader, you just have to click on the *Add Multiples Tags* button :

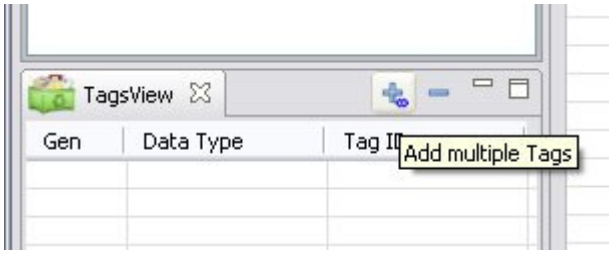


Figure 28 - Add multiples tags in RIFIDI Emulator

The window with the tags parameters looks like the one with the reader parameters :

Figure 29 - Tags parameters

You just have to choose the tag type, the generation and the amount of tags to create. For certain type of type a prefix is needed, that means that all the tag ID will start with this prefix, add the hex chain you want in the corresponding field and then click on *Finish* button.

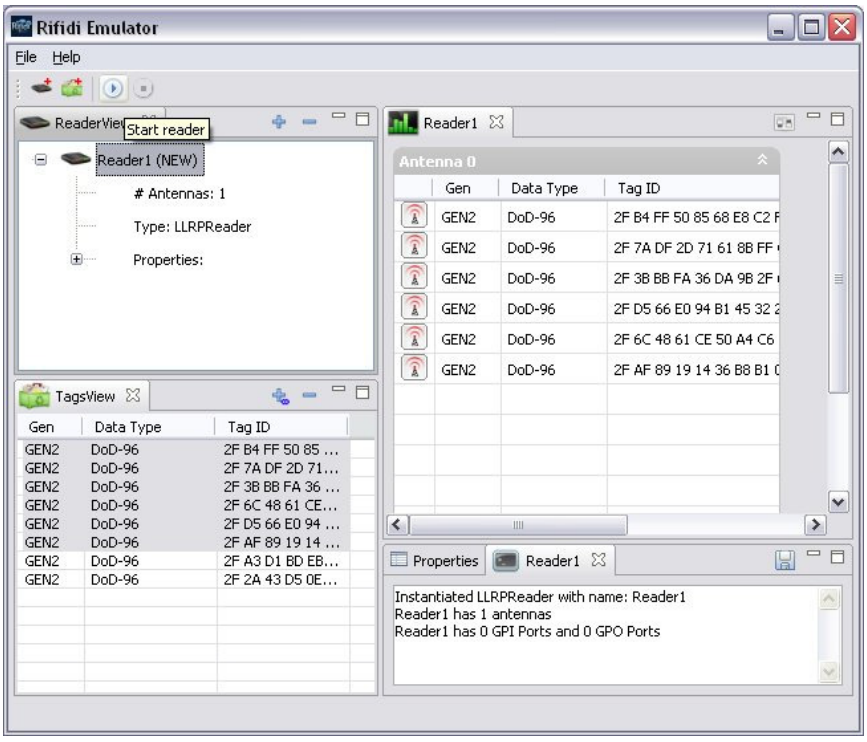


Figure 30 - Start the reader

The configuration is now almost done. The last things to do are select some tags and drag them to the antenna section of the reader. That means that these tags are now readable by the reader. Then select the reader in the *ReaderView* and click on the *Start Reader* button, as on the figure above, your reader is now waiting for connections.

Appendix E – Examples of LLRP messages

GET_READER_CAPABILITIES_RESPONSE

```
<Message type="GetReaderCapabilitiesResponse">
  <LLRPStatus>
    <ErrorCode value="0"/>
    <ErrorDescription value="Success"/>
  </LLRPStatus>
  <GeneralDeviceCapabilities>
    <MaxNumberOfAntennaSupported value="4"/>
    <CanSetAntennaProperties value="false"/>
    <HasUTCClockCapability value="false"/>
    <DeviceManufacturerName value="0"/>
    <ModelName value="0"/>
    <ReaderFirmwareVersion value="LLRP_Reader1.4"/>
    <ReceiveSensitivityTableEntry>
      <Index value="0"/>
      <ReceiveSensitivityValue value="0"/>
    </ReceiveSensitivityTableEntry>
    <GPIOCapabilities>
      <NumGPIs value="0"/>
      <NumGPOs value="0"/>
    </GPIOCapabilities>
    <PerAntennaAirProtocol>
      <AntennaID value="1"/>
      <ProtocolElement value="1"/>
    </PerAntennaAirProtocol>
    <PerAntennaAirProtocol>
      <AntennaID value="2"/>
      <ProtocolElement value="1"/>
    </PerAntennaAirProtocol>
    <PerAntennaAirProtocol>
      <AntennaID value="3"/>
      <ProtocolElement value="1"/>
    </PerAntennaAirProtocol>
    <PerAntennaAirProtocol>
      <AntennaID value="4"/>
      <ProtocolElement value="1"/>
    </PerAntennaAirProtocol>
  </GeneralDeviceCapabilities>
  <LLRPCapabilities>
    <CanDoRFSurvey value="false"/>
    <CanReportBufferFillWarning value="false"/>
    <SupportsClientRequestOpSpec value="false"/>
    <CanDoTagInventoryStateAwareSingulation value="false"/>
    <SupportsEventAndReportHolding value="false"/>
    <MaxPriorityLevelSupported value="0"/>
    <ClientRequestOpSpecTimeout value="0"/>
    <MaxNumROSpecs value="1"/>
    <MaxNumSpecsPerROSpec value="1"/>
    <MaxNumInventoryParameterSpecsPerAISpec value="0"/>
    <MaxNumAccessSpecs value="0"/>
    <MaxNumOpSpecsPerAccessSpec value="0"/>
  </LLRPCapabilities>
  <RegulatoryCapabilities>
    <CountryCode value="0"/>
    <CommunicationsStandard value="0"/>
    <UHFBandCapabilities>
      <TransmitPowerLevelTableEntry>
```

```
            <Index value="0"/>
            <TransmitPowerValue value="0"/>
          </TransmitPowerLevelTableEntry>
          <FrequencyInformation>
            <Hopping value="false"/>
            <FixedFrequencyTable>
              <FrequencyElement value="0"/>
            </FixedFrequencyTable>
          </FrequencyInformation>
          <UHFC1G2RFModeTable>
            <UHFC1G2RFModeTableEntry>
              <ModeID value="0"/>
              <DRValue value="false"/>
              <Conformance value="false"/>
              <Mod value="0"/>
              <FLM value="0"/>
              <M value="0"/>
              <BDRValue value="0"/>
              <PIEValue value="0"/>
              <MinTariValue value="0"/>
              <MaxTariValue value="0"/>
              <StepTariValue value="0"/>
            </UHFC1G2RFModeTableEntry>
          </UHFC1G2RFModeTable>
        </UHFBandCapabilities>
      </RegulatoryCapabilities>
      <C1G2LLRPCapabilities>
        <CanSupportBlockErase value="false"/>
        <CanSupportBlockWrite value="false"/>
        <MaxNumSelectFiltersPerQuery value="0"/>
      </C1G2LLRPCapabilities>
    </Message>
```

ADD_ROSPEC_MESSAGE

```
    <Message type="AddROSpec">
      <ROSpec>
        <ROSpecID value="123"/>
        <Priority value="0"/>
        <CurrentState value="0"/>
        <ROBoundarySpec>
          <ROSpecStartTrigger>
            <ROSpecStartTriggerType value="0"/>
          </ROSpecStartTrigger>
          <ROSpecStopTrigger>
            <ROSpecStopTriggerType value="1"/>
            <DurationTriggerValue value="1000"/>
          </ROSpecStopTrigger>
        </ROBoundarySpec>
        <AISpec>
          <AntennaElement value="0"/>
          <AISpecStopTrigger>
            <AISpecStopTriggerType value="1"/>
            <DurationTrigger value="1000"/>
          </AISpecStopTrigger>
          <InventoryParameterSpec>
            <InventoryParameterSpecID value="1234"/>
            <ProtocolID value="1"/>
          </InventoryParameterSpec>
        </AISpec>
```

```
          <ROReportSpec>
            <ROReportTrigger value="2"/>
            <N value="0"/>
            <TagReportContentSelector>
              <EnableROSpecID value="true"/>
              <EnableSpecIndex value="true"/>
              <EnableInventoryParameterSpecID value="true"/>
              <EnableAntennaID value="true"/>
              <EnableChannelIndex value="true"/>
              <EnablePeakRSSI value="true"/>
              <EnableFirstSeenTimestamp value="true"/>
              <EnableLastSeenTimestamp value="true"/>
              <EnableTagSeenCount value="true"/>
              <EnableAccessSpecID value="true"/>
              <C1G2EPCMemorySelector>
                <EnableCRC value="false"/>
                <EnablePCBits value="false"/>
              </C1G2EPCMemorySelector>
            </TagReportContentSelector>
          </ROReportSpec>
        </ROSpec>
      </Message>
```

RO_ACCESS_REPORT

```
      <Message type="ROAccessReport">
        <TagReportData>
          <EPC96>
            <EPC value="2F 0C 5C 42 38 54 E2 F6 63 9F 44 C3 "/>
          </EPC96>
          <ROSpecID>
            <ROSpecID value="3"/>
          </ROSpecID>
          <SpecIndex>
            <SpecIndex value="0"/>
          </SpecIndex>
          <InventoryParameterSpecID>
            <InventoryParamSpecID value="0"/>
          </InventoryParameterSpecID>
          <AntennaID>
            <AntennaID value="1"/>
          </AntennaID>
          <PeakRSSI>
            <PeakRSSI value="0"/>
          </PeakRSSI>
          <ChannelIndex>
            <ChannelIndex value="0"/>
          </ChannelIndex>
          <FirstSeenTimestampUTC>
            <Microseconds value="1213305450421000"/>
          </FirstSeenTimestampUTC>
          <LastSeenTimestampUTC>
            <Microseconds value="1213306047315000"/>
          </LastSeenTimestampUTC>
          <TagSeenCount>
            <TagCount value="3"/>
          </TagSeenCount>
        </TagReportData>
        <TagReportData>
          <EPC96>
```

```
            <EPC value="2F 33 AD 8A E9 BD 8F 6F 83 3F 71 CB "/>
          </EPC96>
          <ROSpecID>
            <ROSpecID value="3"/>
          </ROSpecID>
          <SpecIndex>
            <SpecIndex value="0"/>
          </SpecIndex>
          <InventoryParameterSpecID>
            <InventoryParamSpecID value="0"/>
          </InventoryParameterSpecID>
          <AntennaID>
            <AntennaID value="1"/>
          </AntennaID>
          <PeakRSSI>
            <PeakRSSI value="0"/>
          </PeakRSSI>
          <ChannelIndex>
            <ChannelIndex value="0"/>
          </ChannelIndex>
          <FirstSeenTimestampUTC>
            <Microseconds value="1213305450421000"/>
          </FirstSeenTimestampUTC>
          <LastSeenTimestampUTC>
            <Microseconds value="1213306047315000"/>
          </LastSeenTimestampUTC>
          <TagSeenCount>
            <TagCount value="3"/>
          </TagSeenCount>
        </TagReportData>
        <TagReportData>
          <EPC96>
            <EPC value="2F 60 DB 59 CB 46 3B E4 5C A9 BB CE "/>
          </EPC96>
          <ROSpecID>
            <ROSpecID value="3"/>
          </ROSpecID>
          <SpecIndex>
            <SpecIndex value="0"/>
          </SpecIndex>
          <InventoryParameterSpecID>
            <InventoryParamSpecID value="0"/>
          </InventoryParameterSpecID>
          <AntennaID>
            <AntennaID value="1"/>
          </AntennaID>
          <PeakRSSI>
            <PeakRSSI value="0"/>
          </PeakRSSI>
          <ChannelIndex>
            <ChannelIndex value="0"/>
          </ChannelIndex>
          <FirstSeenTimestampUTC>
            <Microseconds value="1213305450421000"/>
          </FirstSeenTimestampUTC>
          <LastSeenTimestampUTC>
            <Microseconds value="1213306047315000"/>
          </LastSeenTimestampUTC>
          <TagSeenCount>
            <TagCount value="3"/>
          </TagSeenCount>
```

```
        </TagReportData>
      </Message>
```

References

[1] The RF in RFID – Physical layer operation of passive UHF tags and readers

http://www.enigmatic-consulting.com/Communications_articles/RFID/RFID_protocols.html

[2] Wikipedia English – Radio-Frequency Identification

http://en.wikipedia.org/wiki/Rfid

[3] EPC Global

http://www.epcglobalinc.org/home

[4] EPC Global LLRP – Low Level Reader Protocol

http://www.epcglobalinc.org/standards/llrp

[5] School of Information Technology and Engineering – SITE

http://www.site.uottawa.ca/eng/

[6] Wikipedia English – University of Ottawa

http://en.wikipedia.org/wiki/University_of_Ottawa

[7] Wikipedia English – University of Ottawa – Research

http://en.wikipedia.org/wiki/University_of_Ottawa#Research

[8] Research Proposal *"Development and Simulation of Enhanced RFID Systems"*

Miodrag Bolic, Ata M.Khan, Mustapha C.E. Yagoub, Oliver W.W. Yang

[9] EPC Global - Class 1 Generation 2 UHF Air Interface Protocol

http://www.epcglobalinc.org/standards/uhfc1g2

[10] RIFIDI project

http://rifidi.org

[11] RIFIDI Source code SVN

https://67.207.137.86/svn/rep-external

[12] Multi channel MAC Protocol

http://ieeexplore.ieee.org/iel5/4339774/4339775/04340298.pdf

[13] Opnet Modeler

http://www.opnet.com/solutions/network_rd/modeler.html

[14] LLRP consortium

http://llrp.org

[15] Impinj Speedway reader

http://www.impinj.com/rfid/rfid-reader.aspx?ekmensel=c580fa7b_8_28_168_2